

# PROTOCOLS FOR RELIABLE AND SECURE MESSAGE TRANSMISSION

*A THESIS*

*submitted by*

**ASHISH CHOUDHURY**

*for the award of the degree*

*of*

**DOCTOR OF PHILOSOPHY**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS**

**MAY 2010**

## Thesis Certificate

This is to certify that the thesis entitled **Protocols for Reliable and Secure Message Transmission** submitted by **Ashish Choudhury** to the Indian Institute of Technology Madras, Chennai for the award of the Degree of Doctor of Philosophy is a record of bona-fide research work carried out by him under my supervision and guidance. The contents of this thesis have not been submitted to any other university or institute for the award of any degree or diploma.

Chennai 600036

Research Guide

Date:

(Prof. C. Pandu Rangan)

*Dedicated To*

*To the All Mighty Supreme Personality of Godhead Sri Krsna and  
to My Parents, Who Brought Me to This World*

## Acknowledgements

First of all, I would like to thank my advisor Prof. C.Pandu Rangan who has been having a tremendous influence on my professional development. He introduced me to the world of Algorithms, which motivated me to do research in Theoretical Computer Science, specially in the field of cryptography. I still remember the day when I went to him with a urge to join PhD under him and asked "Sir, will you guide me if I join PhD under you?" and he replied "you are most welcome". That day really changed my career. I never thought that in him, I am going to get a great advisor, a great teacher, a great critic, a great friend and overall a great philosopher. For the past four years, we have developed a special bond, which is much much more than a student-advisor relationship. I have always considered him my father figure and truly speaking, he also never let me down and always treated me like his own pet child. He always provided me with the best facilities, whether it is a research environment (I can never forget the small, separate TCS Lab II, which he allocated to me for doing peaceful research) or whether it is financial help (I can never forget the personal money which he gave me as advance to visit U.S and Canada for attending CRYPTO 2009 and PODC 2009) or whether it is personal advise (he always gave me personal suggestions whenever I required it). I am always going to miss his after leaving IITM. Nevertheless, I will always be indebted to him for whatever he has done for me till now.

Beside my advisor, I also would like to acknowledge the members of my doctoral committee namely, Prof. P. Sreenivasa Kumar, Prof. S. V. Raghavan, Prof. Kamala Krithivasan, Prof. S. A. Choudam and Dr. A. Thangaraj for their encouragement and invaluable suggestions during my doctoral committee meetings. I would specially like to thank Prof. Kamala Krithivasan, who was my advisor during my M. S (by research) at IITM. She always took special interest in my progress and always motivated and encouraged me to do well. I would also like to thank the present and previous head of the department, Prof. C. Siva Ram Murthy and Prof. T. A. Golsalves, respectively for their kind help.

I would like to thank Dr. Tal Rabin, IBM research for her collaboration and lot of insightful remarks on some of our articles. Her critical remarks helped me a lot to improve my writing style and presentations. I can never forget the brain storming session and interaction that we used to have over Internet.

My special thanks to Dr. Kannan Srinathan of IIT Hyderabad, who was my senior at TCS lab. Truly speaking, he is the one who motivated me to do research in secure distributed computing. I have never seen a person like him. His simplicity and innocence is one thing, which I always tried to follow but without any success. For me, he is the best researcher that TCS lab has produced.

I would like to thank Prof. Bimal Roy and Prof. Palash Sarkar of ISI Kolkata for fruitful discussions (both technical as well as personal) that we had during every time we meet for various conferences. Specially I would like to thank Prof. Bimal Roy (Bimal da) who helped me significantly during by visit to Imphal for National level instructional workshop.

My sincere thanks to all of our department office staffs Radhai Madam, Saradha Madam, Balu Sir, Prema madam and Murali for helping me whenever I required. I would also like to thank Saraswati madam and Raghu sir of administrative block for processing my administrative papers quickly, whenever I required it.

My lab mates Esha, Chaya, Billy, Thiru, Sai, Shinku, Amjed, Nisha, Saki, Manila, Kishore, Sobin, Madhu, Preetha madam, KP, Siddharth, Meena, Harini, Balu, Ash-

win, Raghavendra, Naresh, Sharmila and Vivek deserve special acknowledgements. Specially, I would like to thank Ashwin (BV) for few of the collaborative work that we did together. Special mention goes to Billy, Thiru, Sai, Shinku, Nainesh and the pair (Vivek, Sharmi) for their nice company. Also, their cute son and future cryptographer of TCS lab, swithin deserves special acknowledgement.

The gang of close friends, Rima, Tanmoy, Prasun, Pawan, Hima, Tarun, Ashish, Hitesh, Kanchan, Kishore, Priya have stood by me though the ups and downs of my life. Specially, I would like to thank Kanchan, Priya and Bhanu didi, whom I always loved like my own sisters. They always prayed for me during my undergrad and their prayers helped me a lot to achieve what I am today. All my friends at GRKIST, specially Ashish, Shanil, Rochak, Purvik, Shataksh and many others deserve special acknowledgement. The time that I spent with them during my undergrad is simply unforgettable. My close friends Pawan and Prasun at IITM made my stay in IITM a pleasurable and memorable experience. I will never forget about those endless gossips, laughs that we had at our mess table. Cheers to you guys! Your friendship is truly beyond words of acknowledgement.

My sincere acknowledgements to Infosys Technology India for their Ph.D. fellowship and financial assistance during this work. I would also like to thank IARCS for the financial support provided to me during my visit to Japan for attending ICITS 2009.

Now I would like to thank few remarkable people who had, are having and will continue to have a tremendous influence on my life. My parents, Dr. Ajit Choudhury and Chanda Choudhury, for introducing me to the world and bestowing me with all the love of the world. Though my parents do not know too much about research, they somehow always supported me to do research. Salim uncle, who though being not my own uncle, has always loved me like his own son, deserved a very very special acknowledgement. My uncle, Ashit Choudhury from whom I got the first inspiration to do higher studies also holds a special place for me. My aunt Sikha and my cousin Anamika also deserves special acknowledgement. My dearest brother Avinash (Rahul) holds special place for me. Though we have several conflictions over various issues, he always respected me. He will always hold a special place in my life. I would always respect my maternal grand other and maternal grand father, who virtually gave me a new life. Otherwise, after getting the paralysis attack during my childhood, I would have simply lied down in the bed for the rest of my life. The fact that I am normal today is ONLY and ONLY due to their effort. My grandfather, grandmother, maternal uncle and aunt also deserves special acknowledgement for their blessings for me.

My father in law, mother in law and sister in law also deserves special acknowledgement. In them, I got another family. They always treated me like their own son and brother respectively. My father in law, Mr. Ahibhusan Patra deserves special acknowledgement. I have never seen a true karma yogi like him. He always inspired me to do my best.

Now I would like to say something about one of the most special person in my life. She is no other than my sweet wife Arpita Patra, who also happened to be my research colleague. In her, I have found a true friend, a true critic, a true pillar of support, a true philosopher, a true well wisher and above all, a true loving wife. My every moment of research and personal life has been a cherishing experience in her company. Though, we fight and argue a lot everyday over various personal and research relate issues, at the end of day, she is my loving wife. She is the best gift that IIT Madras has gifted to me.

Finally, I would like to thank the Supreme personality of Godhead, Lord Krsna, who created the universe and gave the supreme knowledge of “niskama karmayoga”

(The art of performing prescribed duties without having any attachment to its fruit and without having any material expectations) to mankind (that I am trying my best to implement in my life). I would sincerely like to thank the devotees of ISKCON, for their wonderful lectures on Bhagwad Gita. This helped me a lot to discover the spirituality within me and understand the true principles of life.

## Abstract

Consider the following problem: a *sender*  $\mathbf{S}$  and a *receiver*  $\mathbf{R}$  are part of an unreliable, connected, distributed network. The distrust in the network is modelled by an entity called *adversary*, who has *unbounded computing power* and who can corrupt some of the nodes of the network (excluding  $\mathbf{S}$  and  $\mathbf{R}$ ) in a variety of ways.  $\mathbf{S}$  wishes to send to  $\mathbf{R}$  a message  $m^{\mathbf{S}}$  that consists of  $\ell$  elements, where  $\ell \geq 1$ , selected uniformly from a finite field  $\mathbb{F}$ . The challenge is to design a protocol, such that after interacting with  $\mathbf{S}$  as per the protocol,  $\mathbf{R}$  should output  $m^{\mathbf{S}}$  without any error (*perfect reliability*). Moreover, this hold irrespective of the disruptive actions done by the adversary. This problem is called *reliable message transmission* or RMT in short. The problem of *secure message transmission* or SMT in short requires an additional constraint that the adversary should not get any information about the message what so ever in information theoretic sense (*perfect secrecy*). Security against an adversary with infinite computing power is also known as *non-cryptographic* or *information theoretic* or *Shannon* security and this is the strongest notion of security. Notice that since the adversary has unbounded computing power, we cannot solve RMT and SMT problem by using classical cryptographic primitives such as public key cryptography, digital signatures, authentication schemes, etc as the security of all these primitives holds good only against an adversary having polynomially bounded computing power.

RMT and SMT problem can be studied in various network models and adversarial settings. We may use the following parameters to describe different settings/models for studying RMT/SMT:

1. Type of Underlying Network — *Undirected Graph, Directed Graph, Hypergraph.*
2. Type of Communication — *Synchronous, Asynchronous.*
3. Adversary capacity — *Threshold Static, Threshold Mobile, Non-threshold Static, Non-threshold Mobile.*
4. Type of Faults — *Fail-stop, Passive, Byzantine, Mixed.*

Irrespective of the settings in which RMT/SMT is studied, the following issues are common:

1. **POSSIBILITY:** What are the necessary and sufficient structural conditions to be satisfied by the underlying network for the existence of any RMT/SMT protocol, tolerating a given type of adversary?
2. **FEASIBILITY:** Once the existence of a RMT/SMT protocol in a network is ascertained, the next natural question is, does there exist an efficient protocol on the given network?
3. **OPTIMALITY:** Given a message of specific length, what is the minimum communication complexity (lower bound) needed by any RMT/SMT protocol to transmit the message and how to design a polynomial time RMT/SMT protocol whose total communication complexity matches the lower bound on the communication complexity (optimal protocol)?

In this dissertation, we look into the above issues in several network models and adversarial settings. This thesis reports several new/improved/efficient/optimal solutions, gives affirmative/negative answers to several significant open problems and last but not the least, provides first solutions to several newly formulated problems.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Overview of RMT and SMT . . . . .	2
1.2	Motivation for Studying RMT and SMT . . . . .	3
1.3	RMT, SMT and Its Variants . . . . .	3
1.4	Various Models for Studying RMT and SMT . . . . .	4
1.4.1	Underlying Network: Undirected, Directed and Hypergraph . . .	4
1.4.2	Type of Communication: Synchronous and Asynchronous . . . .	4
1.4.3	Adversary Capacity . . . . .	4
1.4.4	Type of Faults: Failstop, Passive, Byzantine and Mixed . . . . .	5
1.4.5	Type of Security: Perfect and Statistical . . . . .	5
1.5	Taxonomy of Settings for Studying RMT and SMT . . . . .	5
1.6	A Brief Overview of the Existing Results . . . . .	6
1.7	Motivation of Our Work . . . . .	6
1.8	Our Contribution and Models Discussed in this Thesis . . . . .	8
1.9	Organization of the Thesis and Brief Overview of Our Results . . . . .	8
<b>2</b>	<b>Definition and Preliminaries</b>	<b>14</b>
2.1	Definition of Various Type of Corruptions . . . . .	14
2.2	Definition of RMT, SMT and Its Variants . . . . .	14
2.3	Coding Theory Preliminaries . . . . .	16
2.3.1	RS Decoding Algorithm . . . . .	16
<b>I</b>	<b>Results for PRMT and SRMT in Synchronous Network</b>	<b>27</b>
<b>3</b>	<b>PRMT in Undirected Networks Tolerating Static Byzantine Adversary</b>	<b>28</b>
3.1	Underlying Network Model and Adversary Settings . . . . .	28
3.2	Known Results . . . . .	29
3.2.1	Our Contribution . . . . .	30
3.3	Protocols . . . . .	30
3.3.1	Single Phase PRMT Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	30
3.3.2	Increasing the Throughput of Protocol 1-PRMT-Byzantine . . . .	31
3.3.3	Reliably Communicating a Set of Conflicts . . . . .	32
3.3.4	Protocol 3-Optimal-PRMT-Static-Byzantine . . . . .	34
3.4	Concluding Remarks and Open Problems . . . . .	36
<b>4</b>	<b>PRMT in Undirected Networks Tolerating Mobile Byzantine Adversary</b>	<b>38</b>
4.1	Underlying Network Model and Adversary Settings . . . . .	38

4.2	Existing Literature . . . . .	39
4.2.1	Our Contribution . . . . .	39
4.3	Limitations of Matching Technique . . . . .	40
4.4	Protocols . . . . .	40
4.4.1	Union Technique . . . . .	40
4.4.2	Protocol 3-Optimal-PRMT-Mobile-Byzantine . . . . .	42
4.5	PRMT Tolerating Mobile Adversary (in Terms of Rounds) . . . . .	44
4.5.1	Round Based Network and Adversary Settings . . . . .	45
4.5.2	Transmission Graph . . . . .	46
4.5.3	Computing Minimum Number of Rounds for PRMT with $\rho = 1$ . . . . .	48
4.5.4	Finding $r_{min}$ in the Presence of More than $2t_b + 1$ Paths for $\rho = 1$ . . . . .	50
4.5.5	Computing $r_{min}$ for Arbitrary Roaming Speed . . . . .	51
4.5.6	Computing Minimum Number of Rounds for Static Adversary . . . . .	54
4.5.7	Communication Optimal PRMT Protocol in Terms of Rounds . . . . .	55
4.6	Concluding Remarks and Open Problems . . . . .	57
<b>5</b>	<b>On Tradeoff Among Network Connectivity, Phase Complexity and Communication Complexity of PRMT Tolerating Static Mixed Adversary</b> . . . . .	<b>59</b>
5.1	Underlying Network Model and Adversary Settings . . . . .	59
5.2	Existing Results for PRMT Tolerating $\mathcal{A}_{(t_b, t_f)}^{static}$ . . . . .	60
5.2.1	Justification to Study PRMT Tolerating $\mathcal{A}_{(t_b, t_f)}^{static}$ . . . . .	60
5.3	Holy Grail Problem of PRMT Tolerating $\mathcal{A}_{(t_b, t_f)}^{static}$ . . . . .	61
5.4	Our Solution to the Holy Grail Problem . . . . .	62
5.5	An Overview of Our Results in This Chapter . . . . .	62
5.6	Bounds for Single Phase PRMT Tolerating $\mathcal{A}_{(t_b, t_f)}^{static}$ . . . . .	63
5.6.1	Increasing the Throughput of Protocol 1-PRMT-Mixed . . . . .	65
5.6.2	PRMT Based on Both Error Correction and Detection Capability . . . . .	65
5.7	Answer to the Holy Grail Problem of PRMT . . . . .	68
5.8	Upper Bound on Phase Complexity of PRMT Tolerating $\mathcal{A}_{(t_b, t_f)}^{static}$ . . . . .	73
5.8.1	Protocol 3-PRMT-Static-Mixed: A Three Phase PRMT Protocol . . . . .	73
5.8.2	A Worst Case $\mathcal{O}(\mathcal{D})$ Phase PRMT Protocol Tolerating $\mathcal{A}_{(t_b, t_f)}^{static}$ . . . . .	75
5.9	Concluding Remarks and Open Problems . . . . .	79
<b>6</b>	<b>PRMT in Undirected Networks Tolerating Mobile Mixed Adversary</b> . . . . .	<b>81</b>
6.1	Network Model and Adversary Settings . . . . .	81
6.2	Characterization and Lower Bound on Communication Complexity . . . . .	82
6.2.1	PRMT Against $\mathcal{A}_{(t_b, t_f)}^{mobile}$ Requires More Communication Than PRMT Against $\mathcal{A}_{(t_b, t_f)}^{static}$ . . . . .	88
6.3	Communication Optimal PRMT Tolerating $\mathcal{A}_{(t_b, t_f)}^{mobile}$ . . . . .	88
6.4	Concluding Remarks and Open Problems . . . . .	91
<b>7</b>	<b>PRMT in Directed Networks Tolerating Static Byzantine Adversary</b> . . . . .	<b>92</b>
7.1	Directed Network Model . . . . .	92
7.2	Characterization of Communication Optimal PRMT in Directed Network . . . . .	93
7.2.1	Black Box Used in Our Protocol . . . . .	94
7.2.2	A Three Phase Communication Optimal PRMT Protocol . . . . .	96
7.3	Concluding Remarks and Open Problems . . . . .	98

<b>8</b>	<b>SRMT in Undirected Networks Tolerating Static Mixed Adversary</b>	<b>99</b>
8.1	Network Model and Adversary Settings . . . . .	99
8.1.1	Tools Used in SRMT and SSMT Protocols . . . . .	99
8.2	Characterization of SRMT and Bounds on the Communication Complexity of SRMT . . . . .	100
8.2.1	Characterization of SRMT in Undirected Networks . . . . .	100
8.2.2	Lower Bound and Upper Bound for Single Phase SRMT . . . . .	101
8.2.3	Lower Bound and Upper Bound for Multi Phase SRMT . . . . .	105
8.3	Concluding Remarks and Open Problems . . . . .	105
<b>II</b>	<b>Results for PSMT and SSMT in Synchronous Network</b>	<b>107</b>
<b>9</b>	<b>PSMT in Undirected Networks Against Static Byzantine Adversary</b>	<b>108</b>
9.1	Existing Results for PSMT Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	108
9.1.1	Single Phase PSMT Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	108
9.1.2	Multi Phase PSMT Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	109
9.2	A Three Phase Polynomial Time PSMT Protocol . . . . .	111
9.2.1	Extracting Randomness . . . . .	112
9.2.2	Protocol 3-Optimal-PSMT-Static-Byzantine . . . . .	112
9.3	Two Phase Communication Optimal PSMT Protocol . . . . .	114
9.4	Open Problem . . . . .	115
<b>10</b>	<b>PSMT in Undirected Networks Against Mobile Byzantine Adversary</b>	<b>116</b>
10.1	Network Model and Adversary Settings . . . . .	116
10.2	Existing Results for PSMT Tolerating $\mathcal{A}_{t_b}^{mobile}$ . . . . .	117
10.3	A Three Phase Communication Optimal PSMT Tolerating $\mathcal{A}_{t_b}^{mobile}$ . . . . .	117
10.4	Concluding Remarks and Open Problems . . . . .	120
<b>11</b>	<b>PSMT in Undirected Networks Tolerating Static Mixed Adversary</b>	<b>121</b>
11.1	Network Model and Adversary Settings . . . . .	121
11.2	Existing Results for PSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . . . . .	122
11.2.1	Characterization and Lower Bound for Single Phase PSMT . . . . .	122
11.2.2	Characterization and Lower Bound for Multi Phase PSMT . . . . .	123
11.3	Limitations of Protocol 3-Optimal-PSMT-Static-Byzantine Against $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . . . . .	124
11.4	A Four Phase Communication Optimal PSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . . . . .	125
11.4.1	A Conditional Single Phase Protocol to Establish a One Time Pad . . . . .	125
11.4.2	A Three Phase Protocol to Identify at least $\frac{t_b}{2}$ Byzantine Corrupted Wires . . . . .	127
11.4.3	Protocol 4-Optimal-PSMT-Static-Mixed . . . . .	133
11.5	Concluding Remarks and Open Problems . . . . .	135
<b>12</b>	<b>PSMT in Undirected Networks Tolerating Mobile Mixed Adversary</b>	<b>137</b>
12.1	Network Model and Adversary Settings . . . . .	138
12.2	Characterization and Lower Bound on Communication Complexity for PSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . . . . .	138
12.3	Limitations of Protocol 3-Optimal-PSMT-Mobile-Byzantine and 4-Optimal-PSMT-Static-Mixed Against $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . . . . .	138
12.4	Nine Phase Communication Optimal PSMT . . . . .	139
12.5	Three Phase Communication Optimal PSMT Protocol Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . . . . .	145

12.5.1	Pseudo-Basis and Pseudo-Dimension . . . . .	145
12.5.2	How to Find Pseudo-Basis . . . . .	149
12.5.3	Protocol 3-Optimal-PSMT-Mobile-Mixed Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . . .	151
12.6	Concluding Remarks and Open Problems . . . . .	152
<b>13</b>	<b>PSMT in Directed Networks Tolerating Static Byzantine Adversary</b>	<b>155</b>
13.1	Network Model and Adversary Settings . . . . .	155
13.2	Existing Results for PSMT in Directed Network . . . . .	156
13.3	Overview of Our Results for PSMT in Directed Networks . . . . .	156
13.4	Tools Used in Our Protocols . . . . .	157
13.5	Two Phase PSMT in Directed Synchronous Network Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	158
13.5.1	Characterization of Two Phase PSMT in Directed Network . . . . .	158
13.5.2	Lower Bound on Communication Complexity of Two Phase PSMT	162
13.6	Guessing Attack Against PSMT Protocol of [24, 62] in Directed Networks	163
13.6.1	Security of Two Phase PSMT Protocol Against Guessing Attack	166
13.7	New Multi Phase PSMT Protocols Secure Against Guessing Attack . . . . .	166
13.7.1	Three Phase PSMT with $0 < u \leq \frac{t_b}{2}$ and $n = 3t_b - 2u + 1$ . . . . .	166
13.7.2	Six Phase PSMT when $\frac{t_b}{2} < u \leq t_b$ and $n = 2t_b + 1$ . . . . .	169
13.7.3	Six Phase PSMT Protocol when $u > t_b$ and $n = 2t_b + 1$ . . . . .	170
13.8	Lower Bound on Communication Complexity of Multi Phase PSMT . . . . .	170
13.9	Concluding Remarks and Open Problems . . . . .	174
<b>14</b>	<b>SSMT in Undirected Networks Tolerating Static Mixed Adversary</b>	<b>178</b>
14.1	Network Model and Adversary Settings . . . . .	178
14.2	Existing Results for SSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . . . . .	179
14.2.1	Existing Results for Single Phase SSMT . . . . .	179
14.2.2	Existing Results for Multi Phase SSMT . . . . .	183
14.3	A Four Phase Communication Optimal SSMT . . . . .	186
14.3.1	SSMT with Constant Factor Overhead Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	188
14.4	Concluding Remarks and Open Problems . . . . .	188
<b>15</b>	<b>SRMT and SSMT in Directed Networks Tolerating Static Byzantine Adversary</b>	<b>191</b>
15.1	Network Model and Adversary Settings . . . . .	191
15.1.1	Comparison of Our Model with the Model of [84, 72, 80] . . . . .	192
15.2	Existing Results for SRMT and SSMT in Directed Network . . . . .	192
15.3	Overview of Our Results for SRMT and SSMT in Directed Network . . . . .	193
15.3.1	Overview of Our Protocols . . . . .	193
15.3.2	The Roadmap . . . . .	194
15.4	Tools Used in Our SRMT and SSMT Protocols . . . . .	194
15.5	A Three Phase SSMT Protocol in Directed Network Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	194
15.5.1	A Three Phase Exponential SSMT Protocol . . . . .	195
15.5.2	A Three Phase Efficient SSMT Protocol . . . . .	197
15.5.3	Inefficiency of the Three Phase SSMT Protocol of [24, 54, 87] . . . . .	210
15.6	Six Phase Statistically Secure Pad Establishment Protocol . . . . .	214
15.6.1	Single Phase Conditional Pad Establishment Protocol . . . . .	215
15.6.2	A Six Phase Pad Establishment Protocol . . . . .	216
15.7	Communication Optimal SRMT Protocol in Directed Network . . . . .	222
15.7.1	Pre-Processing Step of the Protocol . . . . .	222
15.7.2	First Two Phases of the Protocol . . . . .	223
15.7.3	Phase III and Phase IV of the Protocol . . . . .	225

15.7.4	Remaining Phases of the Protocol . . . . .	230
15.7.5	Summary of the Steps of the Protocol . . . . .	232
15.8	Communication Optimal SSMT Protocol in Directed Network . . . . .	232
15.9	Lower Bound on Communication Complexity of SRMT and SSMT . . . . .	233
15.9.1	Lower Bound for SRMT Protocols . . . . .	233
15.9.2	Lower Bound for SSMT Protocols . . . . .	234
15.10	Concluding Remarks and Open Problems . . . . .	237
<b>III Results for PSMT and SSMT in Asynchronous Network</b>		<b>244</b>
<b>16</b>	<b>PSMT and SSMT in Asynchronous Network Tolerating Static Byzantine Adversary</b>	<b>245</b>
16.1	Asynchronous Network Model . . . . .	246
16.2	Overview of Our Results for PSMT and SSMT in Asynchronous Network	247
16.3	APSMT When All Wires are Unidirectional from <b>S</b> to <b>R</b> . . . . .	248
16.3.1	Existing APSMT Protocol of [69] . . . . .	248
16.3.2	Characterization of APSMT in Presence of Unidirectional Wires	249
16.3.3	APSMT in Presence of $n = 3t_b + 1$ Unidirectional Wires . . . . .	249
16.4	APSMT When All Wires are Bidirectional Between <b>S</b> and <b>R</b> . . . . .	251
16.5	ASSMT When All Wires are Unidirectional from <b>S</b> to <b>R</b> . . . . .	253
16.6	ASSMT When All Wires are Bidirectional Between <b>S</b> to <b>R</b> . . . . .	256
16.7	Concluding Remarks and Open Problems . . . . .	256
<b>17</b>	<b>Conclusion and Directions for Further Research</b>	<b>258</b>
17.1	Summary of Our Contributions . . . . .	258
17.2	Major Inferences Drawn From Our Results . . . . .	259
17.3	Directions for Future Research . . . . .	260

# List of Figures

1.1	The taxonomy of settings in which RMT/SMT can be studied. . . . .	5
2.1	Protocol for RS Decoding . . . . .	25
3.1	Wire Abstraction of a Network. . . . .	29
3.2	Single Phase Reliable Message Transmission Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	31
3.3	Reliably Sending a List of Confliction Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	33
3.4	Three Phase Communication Optimal PRMT Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	35
3.5	Summary of the Results for PRMT in Undirected Synchronous Network Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	37
4.1	Data Flow Over the $n$ Wires During <b>Phase I</b> of Protocol 3-Optimal- PRMT-Mobile-Byzantine . . . . .	41
4.2	Three Phase Communication Optimal PRMT Tolerating $\mathcal{A}_{t_b}^{mobile}$ . . . . .	43
4.3	A $(2t_b + 2)$ - <b>(S,R)</b> -connected Network. . . . .	45
4.4	A 3- <b>(S,R)</b> -connected Network $G$ . . . . .	47
4.5	An $r_{min}$ Round PRMT Protocol on Graph $G^{r_{min}}$ . . . . .	50
4.6	Algorithm for Finding $r_{min}$ in the Presence of More than $2t_b + 1$ Paths for $\rho = 1$ . . . . .	50
4.7	Round-Optimal Reliable Message transmission of $m^S$ Tolerating a Mo- bile Adversary with Roaming Speed of $\rho > 1$ . . . . .	53
4.8	A $3r_{min}$ Round Communication Optimal PRMT Protocol in Terms of Rounds . . . . .	56
4.9	Summary of the Results for PRMT in Undirected Synchronous Network Tolerating $\mathcal{A}_{t_b}^{mobile}$ . . . . .	57
5.1	Single Phase Reliable Message Transmission Tolerating Mixed Adversary	64
5.2	Protocol 2-SP-REL-Mixed . . . . .	66
5.3	A Three Phase PRMT Protocol Tolerating Mixed Adversary . . . . .	74
5.4	An $\mathcal{O}(\mathcal{D})$ Phase PRMT Protocol Tolerating $\mathcal{A}_{(t_b, t_f)}^{static}$ . . . . .	80
6.1	Single Phase PRMTED Protocol . . . . .	85
6.2	A Three Phase Communication Optimal PRMT Protocol Tolerating $\mathcal{A}_{(t_b, t_f)}^{mobile}, n = 2t_b + t_f + 1,  m^S  = nt_b$ . . . . .	89
6.3	Summary of the Results for PRMT in Undirected Synchronous Network Tolerating $\mathcal{A}_{(t_b, t_f)}^{mobile}$ . . . . .	91
7.1	A Single Phase Protocol Based on Error Detection and Error Correction Capability of RS Codes . . . . .	95
7.2	A Three Phase Communication Optimal PRMT Protocol in Directed Network Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	97

8.1	Steps for the Extrapolation Technique . . . . .	102
8.2	The $N \times COL$ Matrix $B_{ext}$ as Constructed from the $ROW \times COL$ Matrix $B_{init}$ . . . . .	102
8.3	Single Phase Protocol 1-Optimal-SRMT-Static-Mixed Tolerating $\mathcal{A}_{(t_b, t_f)}^{static}$ . . . . .	104
8.4	Summary of the Results for SRMT in Undirected Synchronous Network Tolerating $\mathcal{A}_{(t_b, t_f)}^{static}$ . . . . .	106
9.1	Existing PSMT Protocols in Undirected Synchronous Network with $n = 2t_b + 1$ Wires Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	111
9.2	Algorithm for Extracting Randomness . . . . .	112
9.3	A Three Phase Communication Optimal PSMT Protocol Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	113
10.1	Three Phase Communication Optimal PSMT Tolerating $\mathcal{A}_{t_b}^{mobile}$ . . . . .	119
11.1	Single Phase Protocol to Establish a One Time Pad of length $n = 2t_b + t_f + t_p + 1$ . . . . .	126
11.2	A Three Phase Protocol to Identify More than $\frac{t_b}{2}$ Byzantine Faults . . . . .	128
11.3	A Three Phase Protocol to Identify More than $\frac{t_b}{2}$ Byzantine Faults with a Communication Complexity of $\mathcal{O}(n^2)$ . . . . .	132
11.4	A Four Phase Communication Optimal PSMT Protocol Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , $n = 2t_b + t_f + t_p + 1$ , $ m^S  = n$ . . . . .	134
11.5	Summary of the Results for PSMT in Undirected Synchronous Network Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . . . . .	135
12.1	A Nine Phase Communication Optimal PSMT Protocol Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , $n = 2t_b + t_f + t_p + 1$ . . . . .	141
12.2	Remaining Execution of Protocol 9-Optimal-PSMT-Mobile-Mixed if <b>R</b> Executes Step 4 During <b>Phase II</b> . In this case, $ m^S  = n - 1$ . . . . .	142
12.3	Remaining Execution of Protocol 9-Optimal-PSMT-Mobile-Mixed if <b>R</b> Executes Step 5 During <b>Phase II</b> . In this Case, $ m^S  = \frac{n}{2}$ . . . . .	143
12.4	Algorithm to Check Whether $Y$ Can be Linearly Pseudo-Expressed by $\mathcal{B}$ . . . . .	150
12.5	Algorithm to Find the Pseudo-Basis of $\mathcal{Y}$ . . . . .	150
12.6	A Three Phase Communication Optimal PSMT Protocol Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ : $n = 2t_b + t_f + t_p + 1$ , $ m^S  = n^2$ . . . . .	153
12.7	Summary of the Results for PSMT in Undirected Synchronous Network Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . . . . .	154
13.1	A Two Phase Communication Optimal PSMT Protocol Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	160
13.2	Guessing Attack on <b>DW</b> Protocol . . . . .	165
13.3	Three Phase PSMT Protocol: $\ell = n^2u$ , $n = 3t_b - 2u + 1$ , $0 < u \leq \frac{t_b}{2}$ ; $\mathcal{C}$ is the set of all possible RS codewords of length $n$ encoded by polynomials of degree $t_b$ . . . . .	176
13.4	Six Phase PSMT Protocol: $n = 2t_b + 1$ , $\frac{t_b}{2} < u \leq t_b$ , $\ell = n^2u$ ; $\mathcal{C}$ denotes the set of all possible RS codewords of length $N = n + u = 2t_b + 1 + u$ encoded using all possible polynomials of degree $2u > t_b$ . . . . .	177
14.1	Single Phase Communication Optimal SSMT Protocol Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ : $n = 2t_b + t_f + t_p + 1$ , $ m^S  = (t_b + t_f + t_p)$ . . . . .	181
14.2	Three Phase SSMT Protocol Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ : $n = t_b + t_f + \max(t_b, t_p) + 1$ , $ m^S  = 1$ . . . . .	185

14.3	Summary of the Results for SSMT in Undirected Synchronous Network Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . . . . .	188
14.4	Four Phase Communication Optimal SSMT Protocol Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ : $n = t_b + t_f + \max(t_b, t_p) + 1$ . If $t_b \leq t_p$ , then $ m^{\mathbf{S}}  = n^2$ , else $ m^{\mathbf{S}}  = (t_b - t_p)n^2$ . . . . .	190
15.1	Protocol 3-SSMT-Static-Byzantine-Directed-Exponential: A Three Phase Exponential SSMT Protocol, $m^{\mathbf{S}} = \{m_1^{\mathbf{S}}, \dots, m_{\frac{n}{3}}^{\mathbf{S}}\}$ , $ m^{\mathbf{S}}  = \frac{n}{3}$ . . . . .	196
15.2	Phase I of Protocol 3-SSMT-Static-Byzantine-Directed . . . . .	198
15.3	Matrix $T$ as Computed by $\mathbf{S}$ During <b>Phase I</b> of 3-SSMT-Static-Byzantine-Directed . . . . .	199
15.4	Phase II of Protocol 3-SSMT-Static-Byzantine-Directed . . . . .	202
15.5	Phase III of Protocol 3-SSMT-Static-Byzantine-Directed . . . . .	209
15.6	Computation by $\mathbf{S}$ at the End of Phase II of Protocol $\Pi_{modified}^{Existing}$ in [54] . . . . .	211
15.7	Single Phase Protocol 1-Pad . . . . .	216
15.8	First Two Phases of Protocol 6-Pad . . . . .	218
15.9	Remaining Phases of Protocol 6-Pad . . . . .	220
15.10	First Two Phases of Protocol $u$ -Optimal-SRMT-Static-Byzantine-Directed: $ m^{\mathbf{S}}  = (t_b - \frac{u}{2} + 1)n^2$ . . . . .	224
15.11	Execution of $u$ -Optimal-SRMT-Static-Byzantine-Directed If $\mathbf{S}$ Does Not Receive $\frac{u}{2} + 1$ Identical Conflict Lists Through the Bottom Band . . . . .	226
15.12	Local Computation by $\mathbf{S}$ at the End of Phase II if $\mathbf{S}$ Receives $\frac{u}{2} + 1$ Identical Conflict Lists Through the Bottom Band . . . . .	227
15.13	Execution of $u$ -Optimal-SRMT-Static-Byzantine-Directed If $\mathbf{S}$ Receives $\frac{u}{2} + 1$ Identical Conflict Lists and $ L_{fault}^{\mathbf{S}}  \leq (t_b - \frac{u}{2})$ . . . . .	239
15.14	Execution of $u$ -Optimal-SRMT-Static-Byzantine-Directed If $\mathbf{S}$ Receives $\frac{u}{2} + 1$ Identical Conflict Lists and $ L_{fault}^{\mathbf{S}}  \geq (t_b - \frac{u}{2}) + 1$ . . . . .	240
15.15	Execution of $u$ -Optimal-SRMT-Static-Byzantine-Directed If $\mathbf{S}$ Receives "continue" Signal Through the Majority Wires in the Bottom Band at the End of <b>Phase IV</b> . . . . .	241
15.16	Summary of the Steps Executed in Protocol $u$ -Optimal-SRMT-Static-Byzantine-Directed . . . . .	242
15.17	An $\mathcal{O}(u)$ Phase Communication Optimal SSMT Protocol $u$ -Optimal-SSMT-Static-Byzantine-Directed . . . . .	243
16.1	Protocol $\Pi_{APSM T}^{Unidirectional}$ with $n = 3t_b + 1$ unidirectional wires from $\mathbf{S}$ to $\mathbf{R}$ . . . . .	250
16.2	Protocol $\Pi_{ASSM T}^{Unidirectional}$ with $n = 2t_b + 1$ unidirectional wires from $\mathbf{S}$ to $\mathbf{R}$ . . . . .	255
16.3	Connectivity Requirement for PSMT and SSMT in Asynchronous Network Tolerating $\mathcal{A}_{t_b}^{static}$ . . . . .	256

# Chapter 1

## Introduction

Achieving secure communication between two entities who are far away from each other is an age old problem. The notion of "security" was first formalized by Shannon in his classical work [73]. In his work, Shannon assumed the following: there exists a sender **S** and a receiver **R** who are connected by an insecure channel. There exists a *computationally unbounded* adversary who can listen all the communication over the channel. Under these circumstances, Shannon showed that in order to securely send a message  $m^{\mathbf{S}}$  to **R** of size  $\ell$  bits, the following should hold:

1. **S** and **R** should have a *common secret key* of size  $\ell$  bits *beforehand*;
2. The common secret key cannot be established between **S** and **R** using this common channel;
3. Each time **S** wants to send a new message, **S** and **R** should use a new common key. That is, *keys cannot be re-used*.

The security achieved against the computationally unbounded adversary was termed "perfect security" or "perfect secrecy" or unconditional secrecy or information theoretic security. However, the above shortcomings made achieving perfect secrecy impractical. This motivated the research community to look for other alternatives. Two distinct approaches emerged in the literature as practical alternatives:

1. **Polynomially Bounded Adversary:** This approach was first introduced by Diffie and Hellman in their landmark paper [26]. In this approach, it is assumed that the computational power of the adversary is *polynomially* bounded, but the adversary has full access to the cipher text. This models a 'passive' eavesdropper of the channel between **S** and **R**. Under this assumption, Diffie and Hellman showed how **S** and **R** can establish a common secret key using the insecure channel connecting them. In their work, Diffie and Hellman also proposed a new paradigm, called *public-key cryptography* (PKC). In this paradigm, every party holds a public key and a private key, where the private key is kept secret and the public key is available to every one. If **S** wants to send a message then **S** encrypts the message using the public key of **R** and sends it to **R**. **R** then recovers the message by decrypting the encrypted message using his private key.

The work of Diffie and Hellman was followed by that of [67], who proposed a public key cryptosystem, whose security depends upon the hardness assumption of certain number theoretic problem.

2. **Bounded Bandwidth Adversary:** This approach is based on the assumption that in reality, the information obtained by the adversary on the cipher text is partial. Unlike the traditional problem of secure communication, where  $\mathbf{S}$  and  $\mathbf{R}$  are connected directly by a single insecure channel, in a real life scenario,  $\mathbf{S}$  and  $\mathbf{R}$  may be a part of a large, distributed network. Moreover, it may not be possible for the adversary to control the entire network. This is because the network may be very large and even if the attacker possess sufficient computational power, he would need to spend a considerable amount of time before he succeeds in getting hold of the network completely. And usually, a corrupt node/channel can be detected and corrected by the systems administrator in a relatively much lesser amount of time. Thus, the attacker can, at any given instant, control only a part of the network.

The assumption that the adversary has partial access to the network opens a new line of research, which can make achieving perfect security feasible, without restoring to any computational assumptions. Moreover, it also eliminates the unrealistic requirements, implied by Shannon's theorem to achieve perfect security. This line of research lead to the problem of *reliable message transmission* (RMT) and *secure message transmission* (SMT).

RMT and SMT were first introduced by Dolev, Dwork, Waarts and Yung [28]. In their model,  $\mathbf{S}$  and  $\mathbf{R}$  are connected by multiple, disjoint, communication channels, unlike classical model where only one communication channel is assumed. Moreover, it is assumed that a *computationally unbounded adversary* controls only a limited number of channels. Under this scenario, Dolev et al. showed how to achieve perfect security. The work of Dolev et al. has since set off an entire line of research.

This dissertation deals with RMT and SMT problem in several network model and adversary settings. This chapter is structured in the following manner: First, we recall the taxonomy or the framework for RMT and SMT problem. This taxonomy was first discussed in [75]. This provides a comprehensive framework to explore the depth and width of the problem in a unified way. We then briefly discuss about the existing literature of RMT and SMT and their limitations. We then emphasize on our contribution in this thesis and its impact on the literature. This will help to judge the stand that our results hold with respect to the past history and also to understand how our results have advanced the state-of-the-art research of this field. Lastly, we describe the chapter wise organization of this thesis.

## 1.1 Overview of RMT and SMT

In the problem of RMT,  $\mathbf{S}$  and  $\mathbf{R}$  are part of an unreliable, connected, distributed network. The distrust in the network is modelled by an entity called *adversary*, who has *unbounded computing power* and who can corrupt some of the nodes in the network (excluding  $\mathbf{S}$  and  $\mathbf{R}$ ) in a variety of ways.  $\mathbf{S}$  wishes to send to  $\mathbf{R}$  a message  $m^{\mathbf{S}}$  that consists of  $\ell$  elements, where  $\ell \geq 1$ , selected uniformly and randomly from a finite field  $\mathbb{F}$ . The challenge is to design a protocol, such that after interacting with  $\mathbf{S}$  as per the protocol,  $\mathbf{R}$  should output  $m^{\mathbf{S}}$  without any error (*perfect reliability*), inspite of the disruptive actions done by the adversary. The problem of SMT has an additional requirement that the adversary should not get any information about  $m^{\mathbf{S}}$  what so ever in information theoretic sense (*perfect security*). Security against such an adversary, having unbounded computing power is also known as *non-cryptographic* or *information theoretic* or *Shannon security* and this is the strongest notion of security.

## 1.2 Motivation for Studying RMT and SMT

RMT and SMT are fundamental problems in reliable and secure distributed computing. There are basically two motivations for studying RMT and SMT protocols:

1. Many fundamental fault tolerant distributed computing primitives, such as secure multiparty computation (MPC) [89, 37, 13, 17, 66, 8, 9, 11], Byzantine Agreement (BA) [63, 29, 27, 16, 44], Verifiable Secret Sharing (VSS) [18, 13, 66, 36], etc assume that there exists a direct and secure link between every two nodes in the network. This implies that the underlying network graph is a complete graph, which is an unrealistic assumption. In the networks, where  $\mathbf{S}$  and  $\mathbf{R}$  are not adjacent, RMT/SMT protocols help to simulate a virtual reliable/secure link between  $\mathbf{S}$  and  $\mathbf{R}$ . This way, we can simulate a virtual complete network, over which the above fault tolerant primitives can be executed.
2. The second motivation to study SMT is to achieve information theoretic security. The security of all existing public key cryptosystems, digital signatures are based on the unproven hardness assumptions of certain number theoretic problems. However, the increase in computing speed and advent of new computing paradigms like Quantum computing [74] may render these assumptions very weak or useless in practice. But these factors have no effect on information theoretic security which is the strongest notion of security. Thus in a scenario, when existing public key cryptosystems, digital signatures can not provide satisfactory security, SMT protocols may help to provide effective alternative.

## 1.3 RMT, SMT and Its Variants

We now give informal definition of different variants of RMT/SMT. A more formal and rigorous definition will appear in the next chapter.

1. An RMT protocol is called  $\delta$ -reliable, for any  $\delta = 2^{-\Omega(\kappa)}$ , where  $\kappa$  is the error parameter, if at the end of the protocol,  $\mathbf{R}$  correctly outputs  $\mathbf{S}$ 's message, except with error probability  $\delta$ . Moreover, this should hold, irrespective of the behavior of the adversary.
2. An SMT protocol is called  $\epsilon$ -secure, for any  $\epsilon = 2^{-\Omega(\kappa)}$ , where  $\kappa$  is the error parameter, if at the end of the protocol, the adversary does not get any extra information about  $\mathbf{S}$ 's message, except with probability  $\epsilon$ .
3. A message transmission protocol is called  $(\epsilon, \delta)$ -secure, if it is  $\epsilon$ -secure and  $\delta$ -reliable, for some  $\epsilon, \delta > 0$ .
4. An RMT protocol is called *perfectly reliable*, also called as PRMT, if it is 0-reliable.
5. An RMT protocol is called *statistically reliable*, also called as SRMT, if it is  $\delta$ -reliable, for some  $\delta > 0$ .
6. A message transmission protocol is called *perfectly secure*, also called as PSMT, if it is  $(0, 0)$ -secure.
7. A message transmission protocol is called *statistically secure*, also called as SSMT, if it is  $(0, \delta)$ -secure. Such protocols are also called as *almost perfectly secure* protocols [24, 33, 87, 43].

## 1.4 Various Models for Studying RMT and SMT

There are various network settings and adversary model in which RMT and SMT problem can be studied and has been studied in the past. We may use the following parameters to describe different settings/models for studying RMT and SMT:

1. Underlying network;
2. Type of communication;
3. Adversary Capacity;
4. Type of faults; and
5. Type of security.

We now elaborate on each of the above attributes.

### 1.4.1 Underlying Network: Undirected, Directed and Hypergraph

The simplest of the network is undirected network model, where it is assumed that link between every two nodes is bidirectional and hence support both way communication. In directed network model, it is assumed that every communication link has a direction associated with it. Hypergraphs are the most general form of the network, where instead of edges, we have hyperedges. Each hyperedge will have a source node and a set of receiver(s). Any information sent by the source node will be received identically by all the receiver(s) of the hyperedge.

### 1.4.2 Type of Communication: Synchronous and Asynchronous

In a synchronous network, there exists a global clock in the system and so the transmission delay along every edge of the network is bounded. On the other hand, in an asynchronous network, there is no global clock in the system. Thus each link in the network has arbitrary (but finite) delay. The inherent difficulty in designing a protocol over asynchronous network comes from the fact that we *cannot distinguish between a slow sender and a corrupted sender*. Thus, if a receiver node is expecting some message from a sender node and if no message arrives then the receiver cannot distinguish whether the sender node is corrupted and did not sent the message at all or the message is just delayed in the network.

### 1.4.3 Adversary Capacity

The adversary capacity can be further categorized based on the following two properties:

#### 1.4.3.1 Corruption Capacity: Threshold and Non-Threshold

In threshold adversary settings, the number of nodes that can be corrupted by the adversary is bounded by a threshold. On the other, non-threshold adversary setting is a generalization of threshold settings. In non-threshold setting, the adversary is specified by an adversary structure, which is a set of all possible set of nodes that can be potentially corrupted by the adversary. Moreover, each set in the adversary structure may have different size. During the protocol execution, the adversary is permitted to corrupt the nodes of any one arbitrarily chosen set in the adversary structure. It is easy

to see that a threshold adversary is a special type of non-threshold adversary, where the size of each set in the adversary structure is bounded by a threshold.

#### 1.4.3.2 Adversary Behavior: Static and Mobile

If the adversary is static, then it controls the same set of nodes throughout the protocol execution. This is a valid assumption if the protocol is executed for a short period of time. On the other hand, if the adversary is mobile, then the adversary can corrupt different set of nodes during different instances of the protocol. This models a scenario when a protocol is executed for a long duration. Under this circumstances, the nodes which were corrupted in the earlier stage may be rectified, while in the mean time, the adversary may corrupt some other set of nodes. Such a scenario is best modelled by mobile adversary.

#### 1.4.4 Type of Faults: Failstop, Passive, Byzantine and Mixed

The weakest type of corruption is the failstop corruption, where the adversary can simply stop the complete functioning of a node. This type of corruption is called failstop corruption and models the scenario when a node may get crashed due to system failure or natural calamity. Passive corruption means that the adversary has full access to the the computation and communication of the node. However, the adversary cannot make the node to deviate from the protocol. The most powerful type of corruption is Byzantine corruption, where a node is completely under the control of the adversary and may behave arbitrarily during the protocol execution. Lastly, the adversary may simultaneously control different set of nodes in passive, fail-stop and active fashion; such a generalized adversary is called mixed adversary.

#### 1.4.5 Type of Security: Perfect and Statistical

Based on the security level achieved by a protocol, we may have perfect security or statistical security. See Section 1.3 for more details.

### 1.5 Taxonomy of Settings for Studying RMT and SMT

A taxonomy of settings in which RMT and SMT problem can be studied is presented in Fig. 1.1.

Figure 1.1: The taxonomy of settings in which RMT/SMT can be studied.

Underlying Network	Type of Communication	Adversary Capacity	Type of Faults	Type of Security
<i>Undirected Graph</i> <i>Directed Graph</i> <i>Undirected Hypergraph</i> <i>Directed Hypergraph</i>	<i>Synchronous</i> <i>Asynchronous</i>	<i>Threshold Static</i> <i>Threshold Mobile</i> <i>Non-Threshold Static</i> <i>Non-Threshold Mobile</i>	<i>Byzantine</i> <i>Fail-Stop</i> <i>Passive</i> <i>Mixed</i>	<i>Perfect</i> <i>Statistical</i>

For example, one may study SMT over an *undirected synchronous* network thwarting a *threshold static mixed* adversary? In this way, hundreds of different models/settings can be formulated and almost all of them are used in practice. Any RMT/SMT protocol is analyzed by the following four parameters:

1. Connectivity of the underlying network;
2. Number of phases, denoted by  $r$ , taken by the protocol. Here a phase is a communication from  $\mathbf{S}$  to  $\mathbf{R}$  or vice-versa,
3. Communication complexity denoted by  $b$ , which is the total number of field elements communicated by  $\mathbf{S}$  and  $\mathbf{R}$  in the protocol and
4. Amount of computation done by  $\mathbf{S}$  and  $\mathbf{R}$  in the protocol.

Irrespective of the settings in which RMT/SMT is studied, the following issues are common:

1. **POSSIBILITY:** What are the necessary and sufficient structural conditions to be satisfied by the underlying network for the existence of any RMT/SMT protocol, tolerating a given type of adversary?
2. **FEASIBILITY:** Once the existence of a RMT/SMT protocol in a network is ascertained, the next natural question is, does there exist an efficient protocol on the given network?
3. **OPTIMALITY:** Given a message of specific length, what is the minimum communication complexity (lower bound) needed by any RMT/SMT protocol to transmit the message and how to design a polynomial time RMT/SMT protocol whose total communication complexity matches the lower bound on the communication complexity (optimal protocol)?

The above taxonomy and a unified framework for a number of research problems were first discussed in [75]. Different techniques are used to resolve the above issues in different settings. For example, the techniques used in designing optimal RMT/SMT protocols in undirected networks are completely different from the ones used in directed networks.

## 1.6 A Brief Overview of the Existing Results

The RMT and SMT problem were first proposed and solved by Dolev et al. [28]. Dolev et al. considered an undirected synchronous network and assumed that the adversary can corrupt at most  $t_b$  nodes in the network in *Byzantine* fashion. The work of Dolev et al. is followed by several other works, which considered RMT/SMT problem in several network settings and adversarial model. For example, [28, 70, 77, 4, 30, 42] studied PRMT and PSMT in undirected synchronous network, tolerating threshold static Byzantine adversary. In [24, 25, 87, 88], PSMT and SSMT in directed network is studied. In [35, 24] the authors have studied RMT and SMT in Hypergraphs. PSMT in asynchronous network is studied in [69, 76]. Threshold setting is discussed in [28, 42], while non-threshold setting is discussed in [41, 76, 25, 84]. In [82], PSMT in mobile setting is discussed. SSMT is studied in [33, 24, 86, 43, 5].

## 1.7 Motivation of Our Work

Though RMT/SMT and its variants have been studied extensively in the past, there are still several limitations of the existing results, which can be categorized as follows:

1. The issue of POSSIBILITY, FEASIBILITY and OPTIMALITY in the context of RMT and SMT in undirected networks has been completely resolved in [28, 70, 77, 4, 30, 81, 42]. However, all these works assume that the underlying network is *synchronous* and the adversary is a *static* adversary, who can corrupt the nodes *only in Byzantine fashion*. Not too much is known regarding the above issues in other network models and adversarial settings. For example, in a typical large network, certain nodes may be strongly protected and few others may be moderately/weakly protected. An adversary may only be able to fail-stop/eavesdrop a strongly protected node, while he may affect a weakly protected node in Byzantine fashion. Thus, we may capture the abilities of an adversary in a more realistic manner, using three parameters  $t_b, t_f, t_p$  where  $t_b, t_f, t_p$  are the number of nodes, which can be under the influence of the adversary in Byzantine, fail-stop and passive fashion respectively. Also it is better to grade different kinds of disruption done by adversary and consider them separately rather than treating every kind of fault as Byzantine fault as this is an “overkill” and causes an over estimation of the resources required for RMT/SMT<sup>1</sup>. Thus it is justifiable to study RMT/SMT in the context of mixed adversary. Unfortunately, not much is known prior to our work, regarding POSSIBILITY, FEASIBILITY and OPTIMALITY of RMT/SMT and its variants in the context of mixed adversary.
2. In undirected network model, it is assumed that the communication link between any two nodes in the network is bi-directional. However, in practice not every communication channel may admit bi-directional communication. For instance, a base-station may communicate to even a far-off hand-held device but the other way round communication may not be possible. In such a scenario, it is more appropriate to model the underlying network as a directed graph. Motivated by this, Desmedt et al. [24, 87] introduced the problem of RMT/SMT in directed network. However, they only resolved the issue of POSSIBILITY and FEASIBILITY and nothing is known regarding the issue of OPTIMALITY of RMT/SMT and its variants in directed network.
3. If a protocol is executed for a very short duration, then it is appropriate to model the adversary as static, who corrupts the same set of nodes throughout the protocol. However, in many practical scenarios, a protocol may be executed for a longer duration, where **S** and **R** may interact for a long time. In such scenarios, some of the faults which are done in the earlier stages, may be identified and fixed and in the mean time, a hacker may attack some other nodes. Evidently, in such cases the mobile adversary models the fault behavior more appropriately than static adversary. The issue of POSSIBILITY and FEASIBILITY of RMT/SMT tolerating mobile adversary has been resolved in [82]. Unfortunately, nothing is known regarding the issue of OPTIMALITY of RMT/SMT in the context of mobile adversary.
4. In a synchronous network, if **S** (**R**) sends some information along a path, then it is assumed that **R** (**S**) will get the information (possibly corrupted) along the path after a *fixed* interval of time. However, this is a very strong assumption because the delay in the arrival of a single message will affect the overall security of the protocol. A typical large network like the Internet can be modelled more accurately by *asynchronous* networks than synchronous networks. The inherent difficulty in designing a protocol in asynchronous network comes from the fact

---

<sup>1</sup>A formal justification of the last statement will appear in the subsequent chapters.

that we cannot distinguish between a *slow sender* and a *corrupted sender*. Due to this inherent difficulty, RMT/SMT protocols in asynchronous networks seem to very involved. In the literature, prior to our work, not too much is known regarding POSSIBILITY, FEASIBILITY and OPTIMALITY of RMT/SMT and its variants in asynchronous network.

## 1.8 Our Contribution and Models Discussed in this Thesis

In this thesis, we extend the state of the art discussed in the previous section by exploring RMT/SMT and its variants in various network models and adversarial settings. In this thesis, we consider the following network model and adversary settings:

1. Underlying Graph: Undirected as well as directed.
2. Type of Communication: Synchronous as well as asynchronous.
3. Adversary Capacity: Threshold static as well as threshold mobile.
4. Type of Faults: Byzantine as well as mixed.
5. Type of security: perfect as well as statistical.

In the next section, we give the organization of the thesis and also informally summarize the results that will be presented in each chapter.

## 1.9 Organization of the Thesis and Brief Overview of Our Results

In this thesis, we will be studying the RMT/SMT and its variants in different network models and adversarial settings. So instead of describing all network models and adversarial settings at the same place, we feel appropriate to discuss about various models in respective chapters. Accordingly, we also give the literature survey related to each model in the corresponding chapters. We believe that this will increase the readability of this thesis.

The thesis is divided into three parts. The first two parts deal with synchronous network, while the last part deals with asynchronous network. The results related to PRMT and SRMT in synchronous network are given in the first part, while the second part contains the results related to PSMT and SSMT in synchronous network. We now give a brief overview of the results that will be discussed in each chapter.

In the next chapter, we present the definitions used in this thesis. We also present the properties of Reed-Solomon (RS) codes, which will be used throughout this thesis.

In Chapter 3, we study PRMT in *undirected synchronous network* tolerating *threshold static Byzantine adversary*, denoted by  $\mathcal{A}_{t_b}^{static}$ . The adversary  $\mathcal{A}_{t_b}^{static}$  controls the *same* set of  $t_b$  nodes throughout the protocol in Byzantine fashion. In [28], it is shown that PRMT tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff there exists  $n \geq 2t_b + 1$  node disjoint paths, also called as *wires* between  $\mathbf{S}$  and  $\mathbf{R}$ . Moreover, in [81], it is shown that any multiphase (more than two phase) PRMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  must communicate  $\Omega\left(\frac{n\ell}{n-t_b}\right)$  field elements to reliably send a message  $m^{\mathbf{S}}$  containing  $\ell$  field elements. This implies that if  $n = 2t_b + 1$ , then any three or more phase PRMT must communicate  $\Omega(\ell)$  field elements because if  $n = 2t_b + 1$ , then  $n = \Theta(t_b)$ . In [49], the authors have shown the tightness of this bound by designing an  $\mathcal{O}(\log t_b)$  PRMT protocol, which reliably

sends a sufficiently large message containing  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements. The core of Chapter 3 contains a discussion of an improved protocol, in which we significantly reduce the phase complexity of the PRMT protocol of [49]. Specifically, we design a *three* phase PRMT protocol which sends a sufficiently large message containing  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements. This result is based on [58].

In Chapter 4, we study PRMT in *undirected synchronous network* tolerating *threshold mobile Byzantine adversary*, denoted by  $\mathcal{A}_{t_b}^{mobile}$ . The adversary  $\mathcal{A}_{t_b}^{mobile}$  controls *different* set of  $t_b$  nodes in Byzantine fashion, during different phases of the protocol. Intuitively, it seems that the connectivity requirement for PRMT tolerating  $\mathcal{A}_{t_b}^{mobile}$  must be more in comparison to its static counter part  $\mathcal{A}_{t_b}^{static}$ . However surprisingly, in [82], the authors have shown that PRMT tolerating  $\mathcal{A}_{t_b}^{mobile}$  is possible iff there exists  $n \geq 2t_b + 1$  *wires* between  $\mathbf{S}$  and  $\mathbf{R}$ , which is same as the connectivity requirement for any PRMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ . Since  $\mathcal{A}_{t_b}^{mobile}$  is more powerful than  $\mathcal{A}_{t_b}^{static}$ , the lower bound on the communication complexity of PRMT against  $\mathcal{A}_{t_b}^{static}$ , as given in [81], must hold against  $\mathcal{A}_{t_b}^{mobile}$ . This implies that if  $n = 2t_b + 1$ , then any three or more phase PRMT must communicate  $\Omega(\ell)$  field elements to reliably send a message containing  $\ell$  field elements against  $\mathcal{A}_{t_b}^{mobile}$ . However, as far our knowledge is concerned, there is no PRMT protocol tolerating  $\mathcal{A}_{t_b}^{mobile}$ , which satisfies this bound. So in Chapter 4, we design a *three* phase PRMT protocol which sends a sufficiently large message containing  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements, tolerating  $\mathcal{A}_{t_b}^{mobile}$ . This protocol is different from the three phase PRMT protocol against  $\mathcal{A}_{t_b}^{static}$ , having similar properties, which is presented in Chapter 3. Our communication optimal PRMT protocol against  $\mathcal{A}_{t_b}^{mobile}$  gives the following conclusion: *if the adversary does only Byzantine corruption, then mobility of the adversary does not hinder to design a three phase PRMT with a communication complexity of  $\mathcal{O}(\ell)$* . This result is based on [60].

All the existing PRMT protocols abstract the paths between  $\mathbf{S}$  and  $\mathbf{R}$  as wires, neglecting the intermediate nodes in the paths. However, in Chapter 4, we show that this causes significant over estimation of the communication complexity as well as the number of communication rounds required by the protocol. So, we consider the underlying paths in its full form, instead of abstracting them as wires and derive a tight bound on the number of communication rounds required to achieve reliable communication from  $\mathbf{S}$  to  $\mathbf{R}$  tolerating a mobile adversary with arbitrary *roaming speed*. We show how our constant phase PRMT protocol against  $\mathcal{A}_{t_b}^{mobile}$  can be easily adapted to design *round optimal* and *bit optimal* PRMT protocol, provided the network is given as a collection of vertex disjoint paths.

Chapter 5 deals with PRMT in *undirected synchronous network, tolerating threshold static mixed adversary*, denoted by  $\mathcal{A}_{(t_b, t_f)}^{static}$ . The adversary  $\mathcal{A}_{(t_b, t_f)}^{static}$  can control disjoint set of  $t_b$  and  $t_f$  nodes in Byzantine and fail-stop fashion respectively. PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$  was first studied in [75], where the author has justified the study of PRMT tolerating mixed adversary. The author has also given the necessary and sufficient condition for PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ . The characterization shows the existence of more fault tolerance in a network, while the existing results offer no such insight. The main contribution of Chapter 5 is that we study the inherent tradeoff among the three important parameters of PRMT, namely the connectivity requirement  $n$ , phase complexity  $r$  and communication complexity  $b$ . Specifically, we give the answer to the following *Holy Grail* problem of PRMT:

Given an  $n$ -connected undirected synchronous network, under the influence

of  $\mathcal{A}_{(t_b, t_f)}^{static}$  and a value  $b$ , what is the minimum number of phases  $r$  needed to reliably send a message  $m^{\mathbf{S}}$ , where  $|m^{\mathbf{S}}| = \ell$ , such that the total communication complexity of the protocol is  $\mathcal{O}(b)$ ?

We solve the above question by deriving a non-trivial lower bound on  $r$ . Moreover, we show that our lower bound is *asymptotically tight*. To the best of our knowledge, our lower bound is first of its kind and captures the inherent tradeoff among  $n, b, \ell$  and  $r$  simultaneously. This result is based on [6].

In Chapter 6, we focus on PRMT in *undirected synchronous network, tolerating threshold mobile mixed adversary*, denoted by  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ . The adversary  $\mathcal{A}_{(t_b, t_f)}^{mobile}$  can control *different* set of  $t_b$  and  $t_f$  nodes in Byzantine and fail-stop fashion respectively in different phases of the protocol. We show that PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$  is possible iff PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$  is possible. This shows that as far as POSSIBILITY is concerned, mobility of the adversary does not add additional constraints. We then derive the lower bound on the communication complexity of any PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$  and show that the bound is *asymptotically tight*. Our lower bound reveals the following important property: *if the adversary is mixed, then any PRMT protocol against mobile adversary requires more communication, as compared to its static counterpart. Thus if the adversary can do mixed type of corruption, then mobility of the adversary affects OPTIMALITY of the protocols.* This is quite interesting because in Chapter 4 we show that mobility of the adversary has no effect on POSSIBILITY, as well as OPTIMALITY, if the adversary does only *Byzantine* corruption. This result is based on [19].

In Chapter 7, we study PRMT in *directed synchronous network, tolerating threshold static Byzantine adversary*, denoted by  $\mathcal{A}_{t_b}^{static}$ . The condition for the POSSIBILITY of PRMT in undirected graph tolerating  $\mathcal{A}_{t_b}^{static}$ , as given in [28], will also hold for directed graphs. However, nothing is known regarding the OPTIMALITY of PRMT in directed network. We make inroads toward this in Chapter 7. Specifically, the main contribution of Chapter 7 is that we resolve the following question:

*what are the necessary and sufficient structural conditions that the underlying graph should satisfy for the possibility of communication optimal PRMT protocol, which sends a message of size  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements over a directed network / digraph?*

The result of this chapter is based on [57].

In Chapter 8, we study SRMT in *undirected synchronous network, tolerating threshold static Mixed adversary*, denoted by  $\mathcal{A}_{(t_b, t_f)}^{static}$ . The results stated in this chapter are not contribution of this thesis. They are recalled from [75]. However, the main purpose of recalling these results here is to make the thesis self complete. This is because several of these results will be used in the subsequent chapters of the thesis.

Chapter 8 marks the end of first part of the thesis. In Chapter 9, we study PSMT in *undirected synchronous network, tolerating  $\mathcal{A}_{t_b}^{static}$* . From [28], any two or more phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff there exists  $n \geq 2t_b + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ . In [81], the authors have shown that any two or more phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$  must communicate  $\Omega\left(\frac{n\ell}{n-2t_b}\right)$  field elements to securely send a message  $m^{\mathbf{S}}$  containing  $\ell$  field elements. This implies that if  $n = 2t_b + 1$ , then any two or more phase PSMT must communicate  $\Omega(n\ell)$  field elements because if  $n = 2t_b + 1$ , then  $n = \Theta(t_b)$ . In [77], the author presented a two phase PSMT protocol, whose communication complexity satisfies the above bound asymptotically. However, in [4], the authors showed that

the two phase PSMT protocol of [77] does not provide perfect reliability. The authors in [4] also presented a two phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , which asymptotically satisfies the lower bound given in [77]. However, the two phase PSMT protocol of [4] has the following two disadvantages: (a) Both  $\mathbf{S}$  and  $\mathbf{R}$  have to perform exponential amount of computation; (b) The protocol is optimal only if the message is exponentially large. The main contribution of Chapter 9 is a three phase polynomial time, communication optimal PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , which satisfies the lower bound given in [81]. Though our PSMT protocol requires one extra phase, it overcomes the shortcomings of the two phase PSMT protocol of [4]. The result of this chapter is based on [58].

In Chapter 10, we study PSMT in *undirected synchronous network* tolerating *threshold mobile Byzantine adversary*, denoted by  $\mathcal{A}_{t_b}^{mobile}$ . Intuitively, it seems that the connectivity requirement for PSMT tolerating  $\mathcal{A}_{t_b}^{mobile}$  must be more than its static counterpart  $\mathcal{A}_{t_b}^{static}$ . However surprisingly in [82], the authors have shown that PSMT tolerating  $\mathcal{A}_{t_b}^{mobile}$  is possible iff PSMT tolerating  $\mathcal{A}_{t_b}^{static}$  is possible. Since  $\mathcal{A}_{t_b}^{mobile}$  is more powerful than  $\mathcal{A}_{t_b}^{static}$ , the lower bound on the communication complexity of PSMT against  $\mathcal{A}_{t_b}^{static}$ , as given in [81], must hold against  $\mathcal{A}_{t_b}^{mobile}$ . This implies that if  $n = 2t_b + 1$ , then any two or more phase PSMT must communicate  $\Omega(n\ell)$  field elements to securely send a message containing  $\ell$  field elements against  $\mathcal{A}_{t_b}^{mobile}$ . However, to the best of our knowledge, the only known communication optimal PSMT protocol against  $\mathcal{A}_{t_b}^{mobile}$ , achieving the above bound is due to [77], which takes  $\mathcal{O}(t_b)$  phases. The major contribution of Chapter 10 is that we design a *three* phase communication optimal PSMT protocol which sends a sufficiently large message containing  $\ell$  field elements by communicating  $\mathcal{O}(n\ell)$  field elements, tolerating  $\mathcal{A}_{t_b}^{mobile}$ . This protocol is different from the three phase communication optimal PSMT protocol against  $\mathcal{A}_{t_b}^{static}$ , having similar properties, which is presented in Chapter 9. Our communication optimal PSMT protocol against  $\mathcal{A}_{t_b}^{mobile}$  gives the following conclusion: *if the adversary does only Byzantine corruption, then mobility of the adversary does not hinder to design a three phase PSMT with a communication complexity of  $\mathcal{O}(n\ell)$ .* The result of this chapter is based on [60].

In Chapter 11, we study PSMT in *undirected synchronous network* tolerating *threshold static mixed adversary*. However, in addition to Byzantine and failstop corruption, we also consider passive corruption. Thus, the mixed adversary, denoted by  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  corrupts a disjoint set of  $t_b$ ,  $t_f$  and  $t_p$  nodes in Byzantine, failstop and passive fashion respectively throughout the protocol. In [75], the author has shown that any multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is possible iff there exists  $n \geq 2t_b + t_f + t_p + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ . Moreover, any multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  must communicate  $\Omega\left(\frac{n\ell}{n - (2t_b + t_f + t_p)}\right)$  field elements to securely send a message containing  $\ell$  field elements. However, no multi phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  was presented in [75]. We completely resolve the issue of OPTIMALITY of multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  in Chapter 11. Specifically, we present a four phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , which securely sends a message containing  $\ell$  field elements, by communicating  $\mathcal{O}(n\ell)$  field elements over  $n = 2t_b + t_f + t_p + 1$  wires. We also show that the techniques used to design communication optimal PSMT protocol against  $\mathcal{A}_{t_b}^{static}$  cannot be extended in a straight forward manner to design communication optimal PSMT protocol against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . So to design our four phase PSMT protocol against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , we use certain new techniques. The result of this chapter is based on [19].

In Chapter 12, we study PSMT in *undirected synchronous network* tolerating *threshold mobile mixed adversary*, denoted by  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , who corrupts different set of  $t_b, t_f$  and  $t_p$  nodes in Byzantine, failstop and passive fashion respectively during different phases of the protocol. To the best of our knowledge, nothing is known in the literature regarding POSSIBILITY and OPTIMALITY of PSMT in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . In Chapter 12, we completely resolve these issues. The result of this chapter is based on [19].

In Chapter 13, we study PSMT in *directed synchronous network* tolerating *threshold static Byzantine adversary*  $\mathcal{A}_{t_b}^{static}$ . PSMT protocols in directed network tolerating  $\mathcal{A}_{t_b}^{static}$  are presented in [24, 62, 53]. However, recently in [88], the authors showed none of these PSMT protocols tolerating  $\mathcal{A}_{t_b}^{static}$  provide perfect secrecy. This brings forth the question of designing PSMT protocols in directed networks tolerating  $\mathcal{A}_{t_b}^{static}$ . We make inroads in this directions by providing communication optimal PSMT protocols in directed networks tolerating  $\mathcal{A}_{t_b}^{static}$ . The result of this chapter is based on [57, 55].

In Chapter 14, we study SSMT in *undirected synchronous network* tolerating *threshold static mixed adversary*  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . The issue of POSSIBILITY and OPTIMALITY of single phase SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  has been resolved in [75]. Moreover, the author in [75] also resolved the issue of POSSIBILITY of multi phase SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . Furthermore, Srinathan [75] also gave the lower bound on the communication complexity of multi phase SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . However, no multi phase communication optimal SSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  was presented. We completely resolve the issue of OPTIMALITY of multi phase SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  in Chapter 14. The contents of Chapter 14 is based on the results that appeared in [59].

In Chapter 15, we discuss about SRMT and SSMT in *directed synchronous network tolerating threshold static Byzantine adversary*, denoted by  $\mathcal{A}_{t_b}^{static}$ . In [24], the authors have given the necessary and sufficient condition for the existence of any SRMT and SSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ . However, to the best of our knowledge, nothing is known in the literature regarding OPTIMALITY of SRMT and SSMT in directed network tolerating  $\mathcal{A}_{t_b}^{static}$ . The main contribution of Chapter 15 is that we present the lower bound on the communication complexity of SRMT and SSMT in directed network tolerating  $\mathcal{A}_{t_b}^{static}$ . Moreover, we show the tightness of the bound by designing communication optimal SRMT and SSMT protocol in directed network tolerating  $\mathcal{A}_{t_b}^{static}$ . The contents of this chapter is based on the results that appeared in [61]. The main reason for presenting the results for SRMT and SSMT together in the same chapter is that few of the results for SSMT will be used for SRMT. To make the chapter more readable and self contained, we present these results together in the same chapter.

Chapter 15 marks the end of second part of the thesis. In Chapter 16, we study PSMT and SSMT in asynchronous network tolerating  $\mathcal{A}_{t_b}^{static}$ . Unlike synchronous network, PSMT and SSMT has not been studied extensively in asynchronous network. To the best of our knowledge, the only known PSMT protocol in asynchronous network tolerating  $\mathcal{A}_{t_b}^{static}$  is dues to [69]. However, in Chapter 16, we show that the PSMT protocol of [69] does not provide perfect security. We then give the characterization for PSMT and SSMT in asynchronous network tolerating  $\mathcal{A}_{t_b}^{static}$ , thus completely resolving the issue of POSSIBILITY. The most interesting fact brought forth by our characterization is the following: our characterization shows that *asynchrony of the network demands higher connectivity of the network for the existence of PSMT protocols*. On the other hand, *asynchrony of the network does not demand higher connectivity of the network for SSMT protocols*. The contents of Chapter 16 is based on the results that appeared in [20].

In the next chapter, we discuss about the definitions used in this thesis. We also discuss about few black box protocols, which will be used in other chapters.

## Chapter 2

# Definition and Preliminaries

In this chapter, we give the definitions which are used throughout this thesis. We discuss about the properties of Reed-Solomon (RS) codes [45] which will be used extensively in our protocols.

### 2.1 Definition of Various Type of Corruptions

We now define the various type of corruptions that can be done by an adversary.

**Definition 2.1 (Fail-stop Corruption [31])** *A node  $P$  is said to be fail-stop corrupted if the adversary can crash  $P$  at will at any time during the execution of the protocol. But as long as  $P$  is alive,  $P$  will honestly follow the protocol and the adversary will have no access to any internal state of  $P$ .*

**Definition 2.2 (Passive Corruption [31])** *A node  $P$  is said to be passively corrupted if the adversary has full access to the information available/generated at  $P$  and all internal states of  $P$ . However,  $P$  will honestly follow the protocol execution.*

**Definition 2.3 (Byzantine Corruption [31])** *A node  $P$  is said to be Byzantine corrupted if the adversary fully controls the actions of  $P$ . The adversary will have full access to the computation and communication of  $P$  and can force  $P$  to deviate from the protocol in any arbitrary manner.*

### 2.2 Definition of RMT, SMT and Its Variants

Let the message to be transmitted be drawn uniformly and randomly from  $\mathbb{F}$ . We define the *view* of a node  $P_j$ , at any point of the execution of a protocol  $\Pi$  to be the information that  $P_j$  can get from its local input to the protocol (if any), all the messages that  $P_j$  had earlier sent or received, the protocol code executed by  $P_j$  and random coins of  $P_j$ . Let  $\mathcal{A}$  be a *computationally unbounded adversary*, who can corrupt some of the nodes in the network in a variety of ways<sup>1</sup>. The view of  $\mathcal{A}$  at any point of the execution of  $\Pi$  is defined as all the information that  $\mathcal{A}$  can get from the views of all the nodes corrupted by  $\mathcal{A}$  (i.e. all the information that these nodes can commonly compute from their views). For a message  $m^{\mathbf{S}} \in \mathbb{F}$ , any adversary  $\mathcal{A}$  and any protocol  $\Pi$ , let  $\widehat{\Gamma}(\mathcal{A}, m^{\mathbf{S}}, \Pi)$  denote the probability distribution on the view of the adversary  $\mathcal{A}$  at the end of the execution of  $\Pi$ . We now give the following definitions:

---

<sup>1</sup> $\mathcal{A}$  can be static or mobile.  $\mathcal{A}$  may corrupt the nodes in Byzantine or fail-stop or passive or mixed fashion. Moreover,  $\mathcal{A}$  may be either threshold or non-threshold.

**Definition 2.4 (Perfectly Reliable Message Transmission (PRMT))** A protocol  $\Pi$  is said to facilitate perfectly reliable message transmission (PRMT) between  $\mathbf{S}$  and  $\mathbf{R}$  if for any message  $m^{\mathbf{S}}$  drawn from  $\mathbb{F}$  and for every adversary  $\mathcal{A}$ , the following condition is satisfied:

1. *Perfect Reliability:*  $\mathbf{R}$  should receive  $m^{\mathbf{S}}$  correctly, without any error.

**Definition 2.5 (Perfectly Secure Message Transmission (PSMT))** A protocol  $\Pi$  is said to facilitate perfectly secure message transmission (PSMT) between  $\mathbf{S}$  and  $\mathbf{R}$  if for any message  $m^{\mathbf{S}}$  drawn from  $\mathbb{F}$  and for every adversary  $\mathcal{A}$ , the following conditions are satisfied:

1. *Perfect Reliability:* same as in PRMT.
2. *Perfect Secrecy or Perfect Security or Information Theoretic Security:*  $\widehat{\Gamma}(\mathcal{A}, m^{\mathbf{S}}, \Pi) = \widehat{\Gamma}(\mathcal{A}, \widehat{m}^{\mathbf{S}}, \Pi)$ , for every  $\widehat{m}^{\mathbf{S}} \neq m^{\mathbf{S}}$ . That is, the two distributions are identical irrespective of the message transmitted.

Any PSMT protocol is also called as  $(0, 0)$ -SMT protocol, having zero error in both reliability as well in security.

**Definition 2.6 (Statistically Reliable Message Transmission (SRMT))** A protocol  $\Pi$  is said to facilitate statistically reliable message transmission (SRMT) between  $\mathbf{S}$  and  $\mathbf{R}$  if for any message  $m^{\mathbf{S}}$  drawn from  $\mathbb{F}$  and for every adversary  $\mathcal{A}$ , the following condition is satisfied:

1. *Statistical Reliability or  $\delta$ -Reliability:*  $\mathbf{R}$  should receive  $m^{\mathbf{S}}$  correctly, except with error probability  $\delta = 2^{-\Omega(\kappa)}$ , where  $\kappa$  is the error parameter.

**Definition 2.7 (Statistically Secure Message Transmission (SSMT))** A protocol  $\Pi$  is said to facilitate statistically secure message transmission (SSMT) between  $\mathbf{S}$  and  $\mathbf{R}$  if for any message  $m^{\mathbf{S}}$  drawn from  $\mathbb{F}$  and for every adversary  $\mathcal{A}$ , the following conditions are satisfied:

1. *Perfect Secrecy:* same as in PSMT.
2. *Statistical Reliability:* same as in SRMT.

SSMT protocols are also called as  $(0, \delta)$ -SMT protocols or almost perfectly secure message transmission protocols.

Unifying all the above definitions, we give the following definition:

**Definition 2.8 ( $(\epsilon, \delta)$ -SMT)** A protocol  $\Pi$  is said to facilitate  $(\epsilon, \delta)$ -SMT between  $\mathbf{S}$  and  $\mathbf{R}$  if for any message  $m^{\mathbf{S}}$  drawn from  $\mathbb{F}$  and for every adversary  $\mathcal{A}$ , the following conditions are satisfied:

1.  *$\delta$ -Reliability:* same as in SRMT.
2.  *$\epsilon$ -Secrecy:*  $|\widehat{\Gamma}(\mathcal{A}_t^{\text{static}}, m^{\mathbf{S}}, \Pi) - \widehat{\Gamma}(\mathcal{A}_t^{\text{static}}, \widehat{m}^{\mathbf{S}}, \Pi)| = \epsilon$ , for all possible  $\widehat{m}^{\mathbf{S}} \neq m^{\mathbf{S}}$ .

**Definition 2.9 (Communication Optimal Protocol)** Let  $\Pi$  be an  $r$  ( $r \geq 1$ ) phase PRMT/PSMT/SRMT/SSMT protocol which sends a message  $m^{\mathbf{S}}$  containing  $\ell$  ( $\ell \geq 1$ ) field elements by communicating  $\mathcal{O}(b)$  field elements, over a network, tolerating a given adversary  $\mathcal{A}$ . If the lower bound on the communication complexity of any  $r$  phase PRMT/PSMT/SRMT/SSMT protocol to send  $m^{\mathbf{S}}$  over such a network in the presence of  $\mathcal{A}$  is  $\Omega(b)$  field elements, then  $\Pi$  is said to be a communication optimal PRMT/PSMT/SRMT/SSMT protocol to send  $m^{\mathbf{S}}$ .

## 2.3 Coding Theory Preliminaries

A  $(t_b, t_f)$ -error-erasure correcting code is used to reliably send message over a noisy channel. Let  $Ch_{(t_b, t_f)}$  denote a noisy channel, where at most  $t_f$  and  $t_b$  locations of a codeword can be arbitrarily erased and changed respectively during the transmission. A  $(t_b, t_f)$ -block error-erasure correcting code encoding a *message* of  $k$  field elements to a *codeword* of  $N$  field elements is an injective mapping  $\mathcal{C} : \mathbb{F}^k \rightarrow \mathbb{F}^N (N > k)$ , where  $\mathbb{F}$  is the underlying field. The encoding function is used in conjunction with a decoding function  $\mathcal{D} : \mathbb{F}^N \rightarrow \mathbb{F}^k$  with the property that if its input differs from a valid codeword in at most  $t_b$  locations, apart from at most  $t_f$  erasures, then  $\mathcal{D}$  outputs the message corresponding to that codeword. We say that the code corrects  $t_b$  Byzantine errors and  $t_f$  erasures. Clearly, such a decoding function will always exist iff any two valid codewords differ in at least  $2t_b + t_f + 1$  locations [45].

We now define *error correction* and *error detection*.

**Definition 2.10 (Error Correction and Error Detection [68])** *Let  $C$  be a codeword transmitted over  $Ch_{(t_b, t_f)}$  and let  $C'$  be the received vector. By a Byzantine error, we mean the event of changing an entry in codeword  $C$ . The error locations are the indices of the entries in which  $C$  and  $C'$  differs. The task of error correction is to find the error locations and error values in the received vector  $C'$ . On the other hand, error detection means an indication by the decoder that errors have occurred, without attempting to correct them.*

We next present *Singleton Bound* which defines the maximum amount of information which can be reliably send using a  $(t_b, t_f)$ -block error-erasure correcting code.

**Theorem 2.11 (Singleton Bound [45])** *Suppose a sender has generated a  $(t_b, t_f)$ -block error-erasure correcting codeword  $C$  of size  $|C| = N$ , for a message block  $M$  of size  $k$  field elements and sends the codeword  $C$  through  $Ch_{(t_b, t_f)}$ . Let  $C'$  be the received vector of size  $N'$ , where  $N' \geq N - t_f$  and let  $C'$  be different from  $C$  in at most  $t_b$  locations. Then the receiver can reconstruct the message  $M$  from  $C'$  iff  $N \geq 2t_b + t_f + k$ .*

We now give the definition of a special type of  $(t_b, t_f)$ -block error-erasure correcting code called Reed-Solomon (RS) codes which we use in our protocols.

**Definition 2.12 (Reed-Solomon (RS) Codes [45])** *For a message block  $M = (m_1, \dots, m_k)$  over  $\mathbb{F}$ , define ReedSolomon polynomial as  $P_M(x) = m_1 + m_2x + \dots + m_kx^{k-1}$ . Let  $\alpha_1, \alpha_2, \dots, \alpha_N$  denote a sequence of distinct, publicly known and fixed elements from  $\mathbb{F}$ , where  $N > k$ . Then the vector  $C = (c_1, c_2, \dots, c_N)$  is called the Reed-Solomon (RS) codeword of length/size  $N$  for the message block  $M$ , where  $c_i = P_M(\alpha_i)$ , for  $i = 1, \dots, N$ . We denote the length/size of  $C$  by  $|C|$ .*

**Theorem 2.13 ([45])** *RS codes satisfy Singleton Bound.*

### 2.3.1 RS Decoding Algorithm

Berlekamp-Welch algorithm is one of the most simple and efficient RS decoding algorithms existing in the literature. The description of this algorithm can be found in any standard Coding theory book, such as [68, 48, 40]. However, the descriptions of the algorithm, as given in these sources, are in terms of several field and algebraic operations, which is specific to coding theory community. Since the main topic of this thesis

is RMT/SMT, where decoding algorithm is only used as a black-box, in order to avoid too much digression, we take the simple description of the algorithm from [51].

Suppose sender has a message of size  $k$  field elements, which he wants to send reliably over  $Ch_{(t_b, t_f)}$  using RS codes. In order to do so, the sender has to encode the message into an RS codeword of size  $N = 2t_b + t_f + k$  (see Theorem 2.11). So the sender constructs *ReedSolomon* polynomial  $P(x)$  of degree  $k - 1$  and constructs the RS codeword  $C = (c_1, \dots, c_N)$  of length  $N$ , where  $c_i = P(\alpha_i)$ , for  $i = 1, \dots, N$ . Finally, the sender sends the codeword  $C$  to the receiver over  $Ch_{(t_b, t_f)}$ .

We now assume the worst case, where exactly  $t_f$  locations get erased in the codeword  $C$  during its transmission. Although any subset of  $t_f$  locations might get erased, we make a simplifying assumption that the last  $t_f$  locations have been erased. Thus, the receiver will receive a shortened vector  $C' = (c'_1, \dots, c'_{N'})$  of length  $N'$ , where  $N' = N - t_f = 2t_b + k$ . Let  $R(x)$  denote the polynomial of smallest degree passing through the points  $(\alpha_1, c'_1), \dots, (\alpha_{N'}, c'_{N'})$ . It is easy to see that  $R(x)$  will differ from  $P(x)$  in at most  $t_b$  values of  $\alpha_j$ . Notice that the received values  $R(\alpha_j)$ 's may not lie on a  $k - 1$  degree polynomial, due to the presence of  $t_b$  corrupted values. In order to get the original message, the receiver has to construct the polynomial  $P(x)$  from these  $N'$  values of  $R(x)$ . The question is, how the receiver can do so?

Our first observation is that if the receiver can find a polynomial  $P'(x)$  of degree  $k - 1$  that agrees with  $R(x)$  at  $k + t_b$  points, then  $P'(x) = P(x)$ . This is because out of the  $k + t_b$  points, at most  $t_b$  could be corrupted. Therefore,  $P'(x) = P(x)$  for at least  $k$  points. But a polynomial of degree  $k - 1$  is uniquely defined by its values at  $k$  points.

Now the question is: how the receiver can find such a polynomial? The receiver could try to guess where the  $t_b$  errors occurred, but this would take too much time (in fact, it would require exponential time). A very clever polynomial-time algorithm for this problem was invented by Berlekamp and Welch. The main idea is to describe the received polynomial  $R(x)$  (constituted by  $R(\alpha_j)$  values), which because of the errors may not be a  $k - 1$  degree polynomial, as a ratio of polynomials. Let  $e_1, \dots, e_{t_b}$  be the  $t_b$  positions at which errors occurred. Define the *error locator polynomial*  $E(x) = (x - e_1)(x - e_2) \dots (x - e_{t_b})$ . It is easy to see that  $E(x)$  is zero at exactly the  $t_b$  points at which errors occurred. Now observe that the following relation holds:

$$P(x)E(x) = R(x)E(x), \text{ for } x = \alpha_1, \dots, \alpha_{N'}. \quad (2.1)$$

The above equation is true for all  $x$  points at which *no errors* occurred, as  $P(x) = R(x)$  at those points. On the other hand, at all  $x$  points where error occurred,  $E(x) = 0$ . So both the sides of the above equation will be zero at those points.

Now let  $Q(x) = P(x)E(x)$ . Then  $Q(x)$  is a polynomial of degree  $t_b + k - 1$  and is therefore specified by  $t_b + k$  coefficients, which are unknown.  $E(x)$  is a polynomial of degree  $t_b$  and is described by  $t_b + 1$  coefficients. Note that the coefficient of  $x^{t_b}$  in  $E(x)$  is 1. So, there are only  $t_b$  unknown coefficients of  $E(x)$ . Thus, there are total  $(t_b + k) + (t_b) = 2t_b + k = N'$  unknowns here. Moreover, we have  $N'$  received values of  $R(x)$ . So from Equation 2.1, by equating  $Q(x) = R(x)E(x)$ , we can form a system of  $N'$  linear equations in  $N'$  unknowns and solve them. Once the system of equations are solved, we get  $Q(x)$  and  $E(x)$ . From the  $E(x)$  polynomial, we get the locations at which errors occurred. We can then find  $P(x)$  by computing the quotient  $Q(x)/E(x)$ .

If  $N' = k + 2t_b$  and if indeed at most  $t_b$  errors occurred, then the decoding algorithm will correctly output the message. For a complete proof of this fact, see [48, 68, 40].

We now demonstrate the working of the above algorithm with few examples. In these examples, for the ease of presentation, we make the following assumptions:

1. Instead of performing the computations over  $\mathbb{F}$ , we perform the computations over the set of whole numbers. However, the same examples will also work if we perform all the computations over a sufficiently large  $\mathbb{F}$ .
2. Instead of using  $\alpha_1, \dots, \alpha_N$  from  $\mathbb{F}$  for computing RS codeword of length  $N$ , we use  $1, \dots, N$  from the set of whole numbers.

**Remark 2.14** *In all the following examples, we will specify  $k$ , the degree of the polynomial used for encoding as a function of  $t_b$ . This is because in all our PRMT and PSMT protocols,  $k$  will be indeed selected as a function of  $t_b$ . In fact, each of the following examples represents one of the possible cases, which may arise in the context of our PRMT and PSMT protocol. During the description of our PRMT and PSMT protocols, we will show how these examples are related with various such cases.*

**Example 2.15** *Let  $t_b = 1, t_f = 0, k = t_b + 1 = 2$  and  $N = k + 2t_b + t_f = 4$ . Let  $m = (1, 2)$ . So the ReedSolomon polynomial is  $P(x) = 1 + 2x$ . The RS codeword of length four will be  $C = (P(1), \dots, P(4)) = (3, 5, 7, 9)$ . Suppose during the transmission of the codeword, the third location gets corrupted and the receiver receives the vector  $(3, 5, 8, 9)$ . Let  $R(x)$  be the minimum degree polynomial passing through the points  $(1, 3), (2, 5), (3, 8)$  and  $(4, 9)$ . It is easy to see that  $R(x)$  is not a polynomial of degree one. The goal of the decoding algorithm will be to find a polynomial of degree  $k - 1 = 1$ , passing through  $k + t_b = 3$  of the  $R(j)$ 's. It is easy to see that there is exactly one such polynomial, namely the one passing through the points  $(1, 3), (2, 5)$  and  $(4, 9)$ . Since  $t_b = 1$ , the error locator polynomial is*

$$E(x) = (x - e_1)$$

Now  $Q(x) = P(x)E(x)$  will be of degree two. So let

$$Q(x) = Ax^2 + Bx + F$$

For  $x = 1, \dots, 4$ , it holds that

$$Q(x) = R(x)E(x)$$

The above relation implies that

$$\begin{aligned} Ax^2 + Bx + F &= R(x)(x - e_1) \\ \implies Ax^2 + Bx + F + e_1R(x) &= xR(x) \end{aligned}$$

Substituting  $x = 1, \dots, 4$  in the above relation, we get the following system of equations:

$$\begin{aligned} A + B + F + 3e_1 &= 3 \\ 4A + 2B + F + 5e_1 &= 10 \\ 9A + 3B + F + 8e_1 &= 24 \\ 16A + 4B + F + 9e_1 &= 36 \end{aligned}$$

Solving the above system of linear equations, we get  $A = 2, B = -5, F = -3$  and  $e_1 = 3$ . Thus  $Q(x) = 2x^2 - 5x - 3$  and  $E(x) = (x - 3)$ . This implies that error has occurred in the third location. Finally, the algorithm computes  $P(x) = Q(x)/E(x) = 1 + 2x$ . Thus the recovered message is  $(1, 2)$ .

In the above example, the value of  $N, t_b, t_f$  and  $k$  satisfies the inequality given in Theorem 2.11. However, if this is not the case, then anything can happen. We illustrate all possible cases with few examples. These examples will also illustrate the cases which will arise, when we use RS encoding and decoding in the context of PRMT/PSMT. We will then give the formal description of the RS decoding algorithm, along with its properties.

**Example 2.16** Let  $t_b = 2, k = t_b + 1 = 3, t_f = 0$  and  $N = 2t_b + 1 = 5$ . Let  $m = (1, 2, 3)$ . So  $P(x) = 1 + 2x + 3x^2$  and the RS codeword of size five is  $(P(1), \dots, P(5)) = (6, 17, 34, 57, 86)$ . Suppose during the transmission of codeword, only one error occurs, instead of  $t_b = 2$  errors. Let the error occurs at the first location and let the received vector be  $(4, 17, 34, 57, 86)$ .

Although only one error has occurred in the received vector (instead of two), the receiver has no information about this fact. The receiver will think that at most two errors are present in the received vector and would try to correct them. However, from Theorem 2.11, we require  $N' = 7$  in order to correct two errors. Furthermore, with  $N' = 5$  and  $k = 3$ , the decoding algorithm will correctly output the original message, only if one error would have occurred in the received vector, which is the case in this example.

If the receiver applies RS decoding algorithm, assuming the number of errors to be corrected is one, then the algorithm will proceed as follows: the decoding algorithm will try to find a polynomial of degree  $k - 1 = 2$ , passing through  $k + 1 = 4$  of the received points. It is easy to see that the only polynomial passing through four of the received points in this case is the original polynomial  $P(x)$ . Since the algorithm is assuming the number of errors to be one, the error locator polynomial will be

$$E(x) = (x - e_1)$$

Also,  $Q(x) = P(x)E(x)$  will be a polynomial of degree three. So let

$$Q(x) = Ax^3 + Bx^2 + Cx + D$$

By substituting  $x = 1, \dots, 5$  in the equation

$$Q(x) = R(x)E(x),$$

we get the following system of linear equations:

$$\begin{aligned} A + B + C + D + 4e_1 &= 4 \\ 8A + 4B + 2C + D + 17e_1 &= 34 \\ 27A + 9B + 3C + D + 34e_1 &= 102 \\ 64A + 16B + 4C + D + 57e_1 &= 228 \\ 125A + 25B + 5C + D + 86e_1 &= 430 \end{aligned}$$

By solving the above system of equations, we get  $A = 3, B = -1, C = -1, D = -1$  and  $e_1 = 1$ . Thus  $E(x) = (x - 1)$  and  $Q(x) = 3x^3 - x^2 - x - 1$ , indicating that error has occurred in the first location. Moreover,  $P(x) = Q(x)/E(x) = 1 + 2x + 3x^2$ . Thus in this case, the receiver will correctly recover the message.

In the above example, the receiver could recover the original message only because the number of *actual* errors that are present in the received vector is *same* as the number

of errors that the receiver *guessed* and tried to correct. But receiver will not be sure whether the recovered message is correct. Because the decoding algorithm tried to correct only one error, where as two errors could be present in the received vector. If the receiver is sure that exactly one error could be present in the received vector, then he is certain that the output of the algorithm is correct. However, since he is unsure about the exact number of errors in the received vector, the receiver cannot take any guarantee of the output polynomial. In fact, if two errors occur in the transmitted codeword, then the decoding algorithm could end up outputting an incorrect message, as illustrated in the next example.

**Example 2.17** *Suppose in the previous example, exactly  $t_b = 2$  errors occur in the transmitted codeword. Namely, the errors occur in the third and fourth location and let the received vector be  $(6, 17, 28, 39, 86)$ . Notice that the errors are introduced in the codeword in such a way that the two corrupted points, namely  $(3, 28)$  and  $(4, 39)$ , along with the first two correct points, namely  $(1, 6)$  and  $(2, 17)$  lie on the polynomial  $0x^2 + 11x - 5$ . This is possible because the original polynomial  $P(x) = 3x^2 + 2x + 1$  is of degree two and two polynomials of degree two can have same value at two points.*

*Now if the receiver assumes that only one error is present in the received vector and tries to correct it, then the decoding algorithm will proceed as follows: the decoding algorithm will try to find a polynomial of degree  $k - 1 = 2$ , passing through  $k + 1 = 4$  of the received points. In this case, there is only one such polynomial, namely  $0x^2 + 11x - 5$ , passing through the points  $(1, 6)$ ,  $(2, 17)$ ,  $(3, 28)$  and  $(4, 86)$  and hence the decoding algorithm will output this polynomial. Out of the received five points, only three points, namely  $(1, 6)$ ,  $(2, 17)$  and  $(5, 86)$  lie on the original polynomial  $P(x)$ .*

*Since the decoding algorithm assumes the number of errors to be one, the error locator polynomial will be  $(x - e_1)$  and  $Q(x) = Ax^3 + Bx^2 + Cx + D$ . After substituting  $x = 1, \dots, 5$  in the relation  $Q(x) = R(x)(x - e_1)$ , we get the following system of equations:*

$$\begin{aligned} A + B + C + D + 6e_1 &= 6 \\ 8A + 4B + 2C + D + 17e_1 &= 34 \\ 27A + 9B + 3C + D + 28e_1 &= 84 \\ 64A + 16B + 4C + D + 39e_1 &= 156 \\ 125A + 25B + 5C + D + 86e_1 &= 430 \end{aligned}$$

*Solving the above system of equations, we get  $Q(x) = 11x^2 - 60x + 25$  and  $E(x) = (x - 5)$ . This will give  $P(x) = Q(x)/E(x) = 11x - 5$ . Thus the decoding algorithm outputs an incorrect polynomial. Moreover, the algorithm has output fifth location as the error location, even though the fifth location in the received vector represents a correct point on original polynomial  $P(x)$ .*

In the above algorithm, the decoding algorithm outputs an incorrect message due to the following reason: the sender sent the codeword  $(6, 17, 34, 57, 86)$ , corresponding to the polynomial  $3x^2 + 2x + 1$ . From Theorem 2.11, receiver will be able to recover the message *only if one Byzantine error occur during the transmission of the codeword*. However, during the transmission of the codeword, *two errors occur instead of one*. The received vector is  $(6, 17, 28, 39, 86)$ . The errors are introduced in such a way that the received vector  $(6, 17, 28, 39, 86)$  has a *distance*<sup>2</sup> of *one* from the codeword  $(6, 17, 28, 39, 50)$ ,

---

<sup>2</sup>The distance between two vectors is the number of locations at which the two vectors have different components.

corresponding to the polynomial  $0x^2 + 11x - 5$ . Since the decoding algorithm tried to correct one error, it will work as if the transmitted codeword was  $(6, 17, 28, 39, 50)$ , which is received as  $(6, 17, 28, 39, 86)$ , due to the introduction of error at *fifth* location. So it will output fifth location as the error location, even *though it is not an error location*. Moreover, the algorithm will output an incorrect message.

The above example illustrates one of the cases, which occurs, when the actual number of errors in the transmitted codeword is more than the number of errors, which the RS decoding algorithm can correct (as given by Theorem 2.11). However, there may be another case. The *actual* number of errors in the transmitted codeword could be more than the number of errors, which the RS decoding algorithm can correct (as given by Theorem 2.11), such that the decoding algorithm fails to output any *meaningful* polynomial. If this is the case, then the decoding algorithm can simply declare that actual number of errors in the received vector is *more* than the number of errors that the decoding algorithm tried to correct. This case is illustrated by the following example:

**Example 2.18** *Suppose  $t_b = 2, t_f = 0, N = 2t_b + 1 = 5$  and  $k = t_b + 1 = 3$ . Let  $m = (1, 2, 0)$ . So  $P(x) = 1 + 2x$  and the transmitted codeword is  $(3, 5, 7, 9, 11)$ . Suppose two errors are arbitrarily introduced in the first two locations and let the received vector be  $(1, 2, 7, 9, 11)$ . From Theorem 2.11, the receiver can recover the original message from the received vector if only one error occurs in the received vector.*

*If the RS decoding algorithm tries to correct one error in the received vector, then the algorithm will proceed as follows: the algorithm will try to find a polynomial of degree two passing through four of the received points. However, in this case, the errors are introduced in such a way that there exist no polynomial of degree two passing through four of the received points. So the algorithm will not output any meaningful polynomial.*

*Since the decoding algorithm assumes the number of errors to be one, the error locator polynomial will be  $(x - e_1)$  and  $Q(x) = Ax^3 + Bx^2 + Cx + D$ . After substituting  $x = 1, \dots, 5$  in the relation  $Q(x) = R(x)(x - e_1)$ , we get the following system of equations:*

$$\begin{aligned} A + B + C + D + e_1 &= 1 \\ 2A + 4B + 2C + D + 2e_1 &= 4 \\ 27A + 9B + 3C + D + 7e_1 &= 21 \\ 64A + 16B + 4C + D + 9e_1 &= 36 \\ 125A + 25B + 5C + D + 11e_1 &= 55 \end{aligned}$$

*Solving the above system of equations, we get  $Q(x) = -\frac{1}{8}x^3 + \frac{7}{2}x^2 - \frac{79}{8}x + 5$  and  $E(x) = (x - \frac{5}{2})$ . Thus the decoding algorithm outputs  $Q(x)$  and  $E(x)$ , which are not meaningful. The error location as pointed out is not an integer. Moreover,  $Q(x)$  does not divide  $E(x)$ . So the algorithm can declare that more than one error are present in the received vector.*

We now give the summary of the four examples, which illustrates four properties of the RS decoding algorithm, which will be further used in the context of our PRMT/PSMT protocols. We will formalize these properties at the end of this section.

1. In Example 2.15, the receiver knows that at most  $t_b$  errors could be present in the received vector. Moreover, the value of  $k, N', t_b$  and  $t_f$  satisfies the inequality given in Theorem 2.11. Hence the decoding algorithm correctly outputs the

message by finding the  $t_b$  errors. Moreover, the receiver is sure that the output polynomial is correct.

2. In Example 2.16, the receiver knows that at most  $t_b$  errors could be present in the received vector. However, only  $\frac{t_b}{2}$  errors are introduced in the received vector. By substituting  $N' = 2t_b + 1, k = t_b + 1$  and  $t_f = 0$  in the inequality of Theorem 2.11, we find that RS decoding algorithm can correctly output the message only if  $\frac{t_b}{2}$  errors are present in the received vector. Since only  $\frac{t_b}{2}$  errors are introduced in the received vector, the RS decoding algorithm when applied to correct  $\frac{t_b}{2}$  errors, correctly outputs the original message. However, *receiver has no way of knowing that the recovered message is correct as he does not know that indeed  $\frac{t_b}{2}$  errors are present in the received vector.*
3. In Example 2.17, more than  $\frac{t_b}{2}$  errors are introduced in the received vector. However, from Theorem 2.11, RS decoding algorithm can correctly output the message only if  $\frac{t_b}{2}$  errors are present in the received vector. But the actual number of errors in the received vector is more than what can be corrected. Moreover, the errors are introduced in such a way that the received vector has a distance of  $\frac{t_b}{2}$  from another valid codeword  $\hat{C}$  (different from the original codeword which was actually sent by the sender). Since the decoding algorithm is applied to correct  $\frac{t_b}{2}$  errors, the algorithm will output incorrect message, corresponding to  $\hat{C}$ . Moreover, the decoding algorithm outputs at least one correct location in the received vector as the error location. Furthermore, the receiver has no way of knowing that the recovered message is incorrect.
4. In Example 2.18, more than  $\frac{t_b}{2}$  errors are introduced in the received vector. From Theorem 2.11, RS decoding algorithm can correctly output the message only if  $\frac{t_b}{2}$  errors are present in the received vector. But the actual number of errors in the received vector is more than what can be corrected. However, the errors are introduced in such a way that the received vector has a distance of more than  $\frac{t_b}{2}$  from all possible valid codewords. Since the decoding algorithm is applied to correct  $\frac{t_b}{2}$  errors, it fails to output any *meaningful* polynomial. In this case, the receiver simply declares that more than  $\frac{t_b}{2}$  errors are present in the received vector.

In all the previous examples, we have only considered the *error correcting capability* of RS codes as given by Theorem 2.11. However, we can use RS codes to either correct errors or detect errors or simultaneously do the both. The following theorem gives the number of errors which can be corrected and detected by RS codes.

**Theorem 2.19** ([45, 24]) *Let  $C$  be an RS codeword of length  $N$ , corresponding to a message of size  $k$  field elements and let  $C$  be transmitted over  $Ch_{(t_b, t_f)}$ . Let  $C'$  be the received vector of size  $N'$ , where  $N' \geq (N - t_f)$ . Then RS decoding can correct upto  $c$  Byzantine errors in  $C'$  and simultaneously detect additional  $d$  Byzantine errors in  $C'$  iff  $N' - k \geq 2c + d$ , such that  $(c + d) \leq t_b$ .*

Notice that Theorem 2.11 is a special case of Theorem 2.19 because we obtain the former by substituting  $d = 0$  and  $c = t_b$  in the later. Theorem 2.19 states that if we use RS decoding algorithm *only for correcting* errors (i.e.,  $d = 0$ ), then it can correct at most  $\frac{(N' - k)}{2}$  errors. Thus if at most  $\frac{(N' - k)}{2}$  errors are present in the received vector, then the decoding algorithm will correctly find them and recovers the original message.

On the other hand, if we use RS decoding algorithm *only for detecting* errors (i.e.,  $c = 0$ ), then it can detect at most  $(N' - k)$  errors. Thus if at most  $(N' - k)$  errors

are present in the received vector, then the decoding algorithm will sense it and will output an error, indicating that at most  $(N' - k)$  errors are present in the received vector. However, unlike error correction, error detection will not output the locations at which errors are present.

If RS decoding algorithm is used with non-zero values of  $c$  and  $d$  (provided they satisfy the inequalities given in Theorem 2.19), then the algorithm can simultaneously correct and detect errors. In this case, the algorithm will first try to correct  $c$  errors. If the number of errors that are present in the received vector is indeed  $c$ , then the algorithm will correct all of them. Moreover, the algorithm will not detect any additional error and will correctly output the message. On the other hand, if more than  $c$  errors but at most  $(c + d)$  errors are present in the received vector, then the algorithm will detect the additional  $d$  errors (other than the  $c$  errors, which it tried to correct) and will output an error, indicating that more than  $c$  errors are present in the received vector. We illustrate the case of simultaneous correction and detection using RS decoding, with the help of following example. This example also illustrate the cases, which will arise in the context of our PRMT/PSMT protocols, when we use RS decoding for simultaneous error detection and correction.

**Example 2.20** *Let  $t_b = 2, t_f = 0, N = 2t_b + 1 = 5, k = \frac{t_b}{2} + 1 = 2, c = \frac{t_b}{2} = 1$  and  $d = \frac{t_b}{2} = 1$ . Let  $m = (1, 2)$ . So  $P(x) = 1 + 2x$  and the transmitted RS codeword is  $C = (3, 5, 7, 9, 11)$ . Since  $t_f = 0, N' = N = 5$ . Substituting the value of  $N', K, c$  and  $d$  in the inequality of Theorem 2.19, we find that RS decoding algorithm will be able to correct one error and detect one additional error in the received vector.*

*Suppose exactly one error occurs in the received vector, say in the first location. The RS decoding algorithm, when applied to correct  $c = 1$  error, will correctly identify the error. This is because the algorithm will try to find a polynomial of degree  $k - 1 = 1$ , passing through  $k + c = k + \frac{t_b}{2} = 3$  of the received points. In this case, there is only one polynomial of degree one, passing through three of the received points, namely the original polynomial  $P(x)$ . So the algorithm will correctly output the polynomial  $P(x)$ . Moreover, the receiver will be sure that the output polynomial is correct because in this case,  $(c + d) = t_b$  and the maximum number of errors that could be present in the received vector is also  $t_b$ . Since the algorithm has not detected any additional error (other than  $c$  errors), it implies that the output polynomial is indeed correct.*

*On the other hand, suppose that more than  $c = \frac{t_b}{2} = 1$  errors are present in the received vector; i.e., suppose two errors are present in the received vector. Moreover, the errors are introduced in the first two locations and let the received vector be  $(5, 6, 7, 9, 11)$ . Notice that here the errors are introduced in such a way that first  $t_b + (k - 1) = 3$  points in the vector, namely  $(1, 5), (2, 6)$  and  $(3, 7)$  lie on polynomial  $x + 4$ . On the other hand, the last  $N' - t_b = 3$  points in the vector, namely  $(3, 7), (4, 9)$  and  $(5, 11)$  lie on polynomial  $2x + 1$ . If we apply the RS decoding algorithm to correct  $c = 1$  error, then the decoding algorithm will try to find a unique polynomial of degree  $k - 1 = 1$ , passing through  $k + c = k + \frac{t_b}{2} = 3$  of the received points. In this case, there are two polynomials of degree one, passing through three of the received points. So the decoding algorithm will output an error. More specifically,  $E(x) = (x - e_1)$  and  $Q(x) = P(x)E(x) = Ax^2 + Bx + F$ . By substituting  $x = 1, \dots, 5$  in the relation*

$Q(x) = R(x)E(x)$ , we get the following system of linear equations:

$$\begin{aligned} A + B + F + 5e_1 &= 5 \\ 4A + 2B + 1F + 6e_1 &= 12 \\ 9A + 3B + 1F + 7e_1 &= 21 \\ 16A + 4B + 1F + 9e_1 &= 36 \\ 25A + 5B + 1F + 11e_1 &= 55 \end{aligned}$$

However, the above system of equations does not have any solution and hence the algorithm will not output any polynomial. This will indicate to the receiver that more than  $c$  errors are presented in the received vector, which are detected by the algorithm.

In the above example, we have considered the case, when  $c + d = t_b$ . If  $c + d < t_b$ , then again anything can happen. For example, the errors could be introduced in such a way that the received vector could have a distance of  $c$  from another valid codeword (other than the one sent by the sender, as in Example 2.17). In this case, the algorithm will output the incorrect polynomial corresponding to the other codeword. Moreover, the receiver will have no way of knowing that the output polynomial is incorrect, as  $(c + d) < t_b$ . On the other hand, the errors could be introduced in such a way that the received vector has a distance of more than  $c$  from all valid codewords (as in Example 2.18). In this case, the algorithm will fail to output any polynomial, indicating to the receiver that more than  $c$  errors are present in the received vector.

We now give the formal description of RS decoding algorithm. In the algorithm, all the computations are performed over  $\mathbb{F}$ . The algorithm takes the following inputs:

1. A vector  $C'$  of length  $N'$ , received over  $Ch_{(t_b, t_f)}$ . Let  $i_1, \dots, i_{N'} \in \{1, \dots, N\}$  denote the indices of the components of the received vector. This implies that the components of the received vector at the remaining indices in the set  $\{1, \dots, N\} - \{i_1, \dots, i_{N'}\}$  are erased. Here  $N$  is the length of the original RS codeword and  $N' \geq (N - t_f)$ . We denote the values in the received vector as  $R(\alpha_{i_1}), \dots, R(\alpha_{i_{N'}})$ .
2. Parameter  $k$ , where  $k - 1$  is the degree of the original polynomial  $P(x)$ , used for encoding the message.
3. Parameters  $c \geq 0$  and  $d \geq 0$ , subject to the condition that  $N' - k \geq 2c + d$  and  $(c + d) \leq t_b$ . Here  $c$  is the number of errors that the algorithm tries to correct and  $d$  is the number of additional errors that the algorithm tries to detect.

The algorithm is formally given in Fig. 2.1.

**Definition 2.21 (Good/Bad Error List)** We call an error list generated by RS-DEC algorithm as “good” if each of the values in the error list, pointed as a corrupted value, is indeed corrupted. Otherwise we call the error list as “bad”. **When an error list is “bad”, it points a correct value in  $C'$  as corrupted.**

We now state few important properties of RS decoding, which will be used in the context of our PRMT/PSMT protocols. We have already illustrated all these properties with examples and hence we will not give formal proof of these properties. For a complete formal proof, we refer [40, 68]. In all these properties, we assume that  $t'_b \leq t_b$  is the actual number of errors that are present in the received vector. The receiver has no information about  $t'_b$ , except that  $t'_b \leq t_b$ .

Figure 2.1: Protocol for RS Decoding

**Algorithm RS-DEC**( $N', C', c, d, k$ )

**Goal: To Find a Polynomial of Degree  $k-1$  Passing Through  $k+c$  Received  $R(\alpha_j)$ 's**

1. Let the *error locator* polynomial be  $E(x) = (x - e_1) \dots (x - e_c)$ . The coefficient of  $x^c$  in  $E(x)$  will be one.
2. Let  $Q(x) = P(x)E(x) = a_{c+k-1}x^{c+k-1} + a_{c+k-2}x^{c+k-2} + \dots + a_0$  be the polynomial of degree  $c + k - 1$ .
3. Form a system of  $N'$  equations, involving  $2c + k \leq N'$  unknowns  $e_1, \dots, e_c, a_{c+k-1}, \dots, a_0$ , by substituting  $x = \alpha_{i_1}, \dots, \alpha_{i_{N'}}$  in the relation  $Q(x) = R(x)E(x)$ .
4. Solve the above system of equations. Now there are the following cases:
  - (a) If the above system of equations fails to give any solution, then output an error. In this case, the receiver concludes that more than  $c$  errors are present in the received vector.
  - (b) If the above system of equations gives a solution, such that the value of at least one of the unknowns  $e_1, \dots, e_c, a_{c+k-1}, \dots, a_0$  is outside the field  $\mathbb{F}$ , then output an error. In this case, the receiver concludes that more than  $c$  errors are present in the received vector.
  - (c) If the above system of equations gives a solution, such that value of at least two distinct unknowns in the set  $\{e_1, \dots, e_c\}$  are same, then output an error. In this case, the receiver concludes that more than  $c$  errors are present in the received vector.
  - (d) If the above system of equations gives a solution, such that value of all the  $2c+k$  unknowns are from the field  $\mathbb{F}$  and each of unknowns in  $\{e_1, \dots, e_c\}$  have distinct values, then do the following:
    - i. Compute  $P(x) = Q(x)/E(x)$ . Let  $P(x) = b_0 + b_1x + \dots + b_{k-1}x^{k-1}$ .
    - ii. Output  $(b_0, \dots, b_{k-1})$  as the message. In addition, output an *error list*, denoted by *Error\_List*. The *Error\_List* indicates the values which are identified to be corrupted in  $C'$ . The *Error\_List* will contain  $c$  pairs. For  $j = 1, \dots, c$ , the  $j^{\text{th}}$  entry of *Error\_List* is of the form  $(e_j, C'_{i_{e_j}})$ , where  $C'_{i_{e_j}}$  denotes the  $i_{e_j}^{\text{th}}$  entry in  $C'$ .

**Property 2.22** *If  $c + d = t_b$  and  $t'_b \leq c$ , then the algorithm will correct all these errors and will detect no additional errors. So the algorithm will output  $P(x)$ , which is the original/correct  $k - 1$  degree polynomial and *Error\_List*, which is a "good" error list (of cardinality at most  $c$ ). Moreover, the receiver is certain that the output polynomial  $P(x)$  is correct and the error list *Error\_List* is "good". This property is illustrated in Example 2.20.*

**Property 2.23** *If  $c + d = t_b$  and  $t'_b > c$ , then the algorithm will fail to output any message, thus indicating to the receiver that more than  $c$  errors are present in the*

received vector. This is because even though the actual number of errors  $t'_b$  is more than  $c$  (which is the number of errors which the algorithm tried to correct), the algorithm has the capability to detect  $(t_b - c) \geq (t'_b - c)$  additional errors. However, the algorithm can only detect the additional errors, but will not be able to correct them. So the algorithm will not output any message. In this case, the receiver concludes that more than  $c$  errors are present in  $C'$ . This property is illustrated in Example 2.20.

**Property 2.24** If  $c + d < t_b$  and  $t'_b \leq c$ , then the algorithm will correct all the  $t'_b$  errors and will correctly output  $P(x)$ . Moreover, *Error\_List* will be a "good" error list (of cardinality at most  $c$ ). However, receiver will not be sure/certain that the output polynomial  $P(x)$  is correct and the error list *Error\_List* is "good". This is because the extra detection capability of the algorithm in this case is less than  $t_b - c$  and the value of  $t'_b$  is unknown to the receiver. This property is illustrated in Example 2.16.

**Property 2.25** If  $c + d < t_b$  and  $t'_b > c$ , such that the received vector has a distance of  $c$  from another valid codeword (different from the one, which was originally sent by the sender), then the algorithm will output the incorrect  $P'(x) \neq P(x)$ , corresponding to the other codeword. Moreover, the *Error\_List* will be "bad" of cardinality at most  $c$ . Furthermore, receiver will not be sure/certain that the output polynomial  $P'(x)$  is correct and the error list *Error\_List* is "bad". This is because the extra detection capability of the algorithm in this case is less than  $t_b - c$  and the value of  $t'_b$  is unknown to the receiver. This property is illustrated in Example 2.17.

**Property 2.26** If  $c + d < t_b$  and  $t'_b > c$ , such that the received vector has a distance of more than  $c$  from all valid codewords, then the algorithm will output error. In this case, the receiver is certain that more than  $c$  errors are present in the received vector. This property is illustrated in Example 2.18.

In the next chapter, we discuss about PRMT in undirected synchronous network, tolerating threshold static Byzantine adversary.

## Part I

# Results for PRMT and SRMT in Synchronous Network

## Chapter 3

# PRMT in Undirected Networks Tolerating Static Byzantine Adversary

In this chapter, we study PRMT<sup>1</sup> in *undirected synchronous network, tolerating threshold static Byzantine adversary*. We first give the formal specification of the network model and adversary settings used in this chapter. We then give the existing results and motivation of our work. We then present our result.

### 3.1 Underlying Network Model and Adversary Settings

In this chapter, we consider the following settings: We assume that the underlying network is a connected synchronous network, represented by an undirected graph, where  $\mathbf{S}$  and  $\mathbf{R}$  are two *non-adjacent* nodes of the graph. All the edges in the network are reliable and secure but the nodes can be corrupted. We assume that there exists a threshold static adversary, denoted by  $\mathcal{A}_{t_b}^{static}$ , who has *unbounded computing power* and who controls at most  $t_b$  nodes (excluding  $\mathbf{S}$  and  $\mathbf{R}$ ) in Byzantine fashion.

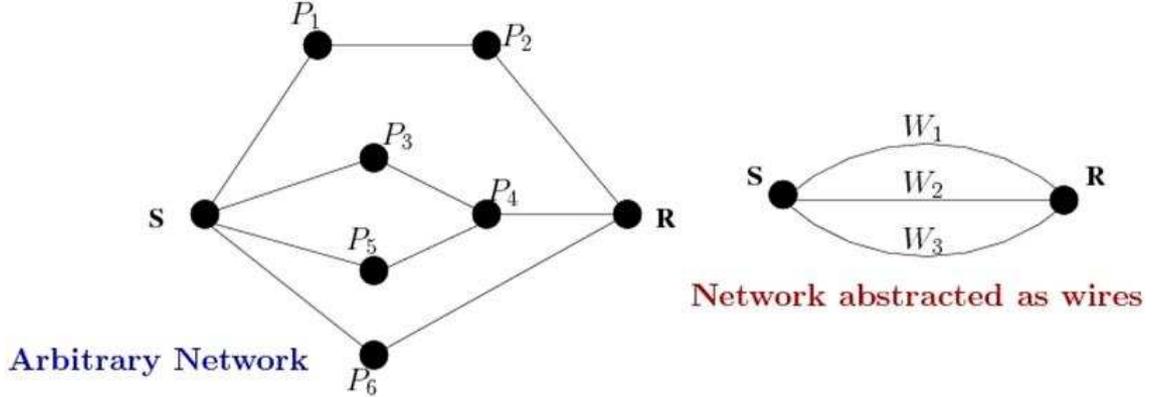
Following the approach of Dolev et al. [28], we abstract away the network and assume that  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n$  parallel and synchronous bi-directional node disjoint paths/channels  $w_1, w_2, \dots, w_n$ , also called as *wires*. The reason for such an abstraction is as follows: suppose some intermediate node between  $\mathbf{S}$  and  $\mathbf{R}$  is under the control of the adversary. Then all the paths between  $\mathbf{S}$  and  $\mathbf{R}$  which passes through that node are also compromised. Hence, all the paths between  $\mathbf{S}$  and  $\mathbf{R}$  passing through that node can be modelled by a single wire between  $\mathbf{S}$  and  $\mathbf{R}$  and we can declare that the wire is corrupted. In the worst case, the adversary can compromise an entire wire in Byzantine fashion by controlling a single node on the wire. Hence  $\mathcal{A}_{t_b}^{static}$ , having unbounded computing power can corrupt up to  $t_b$  wires in Byzantine fashion. A Byzantine corrupted wire may deliver correct information or it may deliver incorrect/changed information. However, in any case, the adversary will completely know the actual information that was sent through a Byzantine corrupted wire. An example of wire abstraction of the network is given in Fig. 3.1.

The set of wires which  $\mathcal{A}_{t_b}^{static}$  controls is decided before the execution of the protocol. Before the execution of the protocol, neither  $\mathbf{S}$  nor  $\mathbf{R}$  knows in advance which wires are going to be influenced by  $\mathcal{A}_{t_b}^{static}$ . However, the total number of wires that can be

---

<sup>1</sup>Recall that in PRMT we require only reliability. So adversary may obtain any information related to the message.

Figure 3.1: Wire Abstraction of a Network.



under the control of  $\mathcal{A}_{t_b}^{static}$  throughout the protocol is bounded by threshold  $t_b$ . Also once a wire is under the control of  $\mathcal{A}_{t_b}^{static}$ , it will remain so throughout that particular execution of the protocol.

We use  $m^{\mathbf{S}}$  to denote the message that  $\mathbf{S}$  wants to send to  $\mathbf{R}$ , where  $m^{\mathbf{S}}$  is a sequence of  $\ell$  field elements, where  $\ell \geq 1$ , selected uniformly from a finite field  $\mathbb{F}$ . Moreover, we assume that all computations and communication in our protocols are done over  $\mathbb{F}$ . The only restriction on  $\mathbb{F}$  is that  $|\mathbb{F}| > n$ . We use  $|m^{\mathbf{S}}|$  to denote the number of field elements in  $m^{\mathbf{S}}$ . We say that a wire is **corrupted**, if the information sent over the wire is changed. A wire which is not under the control of the adversary is said to be **honest**.

In our protocol, we assume the following: if  $\mathbf{S}$  ( $\mathbf{R}$ ) is expecting some information in a specific form along a wire and if no/syntactically incorrect information comes, then  $\mathbf{S}$  ( $\mathbf{R}$ ) assumes some pre-defined value and carry on the computation. Thus we do not consider the case when no information or syntactically incorrect information is received along a wire.

**Definition 3.1 (Broadcast)** *If some information is sent over all the wires then it is said to be “broadcast”. If  $x$  is “broadcast” over at least  $2t_b + 1$  wires, then at most  $t_b$  wires may deliver incorrect  $x$ . But at least  $t_b + 1$  wires will deliver correct  $x$ . So receiver will be able to correctly recover  $x$  by taking majority among the received values.*

We now summarize the existing literature and our result for PRMT in undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{static}$ .

## 3.2 Known Results

PRMT problem was first introduced by Dolev et al. [28], who gave the following characterization:

**Theorem 3.2 ([28])** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n$  wires. Then for any  $r$  ( $r \geq 1$ ), an  $r$ -phase PRMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff  $n \geq 2t_b + 1$ .*

The lower bound on the communication complexity of PRMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  is derived in [77] and in [81]. We now state these lower bounds.

**Theorem 3.3** ([77, 81]) *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + 1$  wires and let  $r \leq 2$ . Then any  $r$ -phase PRMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  must communicate  $\Omega\left(\frac{n\ell}{n-2t_b}\right)$  field elements to reliably send a message containing  $\ell$  field elements.*

**Theorem 3.4** ([81]) *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + 1$  wires and let  $r \geq 3$ . Then any  $r$ -phase PRMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  must communicate  $\Omega\left(\frac{n\ell}{n-t_b}\right)$  field elements to reliably send a message containing  $\ell$  field elements.*

The lower bound given in Theorem 3.3 is tight. Specifically, let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + 1$  wires, which is the minimum number of wires required for any PRMT protocol against  $\mathcal{A}_{t_b}^{static}$ . Then  $\mathbf{S}$  can reliably send a message  $m^{\mathbf{S}}$  containing  $\ell$  field elements by simply broadcasting it. This will take one phase and will require a communication complexity of  $\mathcal{O}\left(\frac{n\ell}{n-2t_b}\right) = \mathcal{O}(n\ell)$  field elements.

If  $n = 2t_b + 1$ , then  $n = \Theta(t_b)$  and hence from Theorem 3.4, any three or more phase PRMT must communicate  $\Omega\left(\frac{n\ell}{n-t_b}\right) = \Omega(\ell)$  field elements to reliably send a message containing  $\ell$  field elements. Moreover, in [49], the authors have shown that this bound is asymptotically tight. Specifically, the authors have shown the following:

**Theorem 3.5** ([49]) *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + 1$  wires. Then there exists an  $\mathcal{O}(\log t_b)$  phase PRMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , which reliably sends a message containing  $\ell = \Theta(n \log^2 n)$  field elements by communicating  $\mathcal{O}(\ell)$  field elements.*

Though the protocol given in [49] is communication optimal, it requires too many phases. This motivates us to design a communication optimal PRMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  with less number of phases.

### 3.2.1 Our Contribution

We significantly improve the phase complexity of the PRMT protocol of [49]. Specifically, we design a three phase PRMT protocol called 3-Optimal-PRMT-Static-Byzantine with  $n = 2t_b + 1$  wires, which reliably sends a message containing  $\ell = \Theta(n^2)$  field elements by communicating  $\mathcal{O}(n^2)$  field elements. Thus, we get reliability with constant factor overhead and that too in constant phases. From Theorem 3.4, our protocol is *phase optimal*. Moreover, from Theorem 3.3, our protocol requires minimum connectivity. It should be noted that our protocol achieves optimality only if  $\ell = \Theta(n^2)$ , while the protocol of [49] achieves it for  $\ell = \Theta(n \log^2 n)$ .

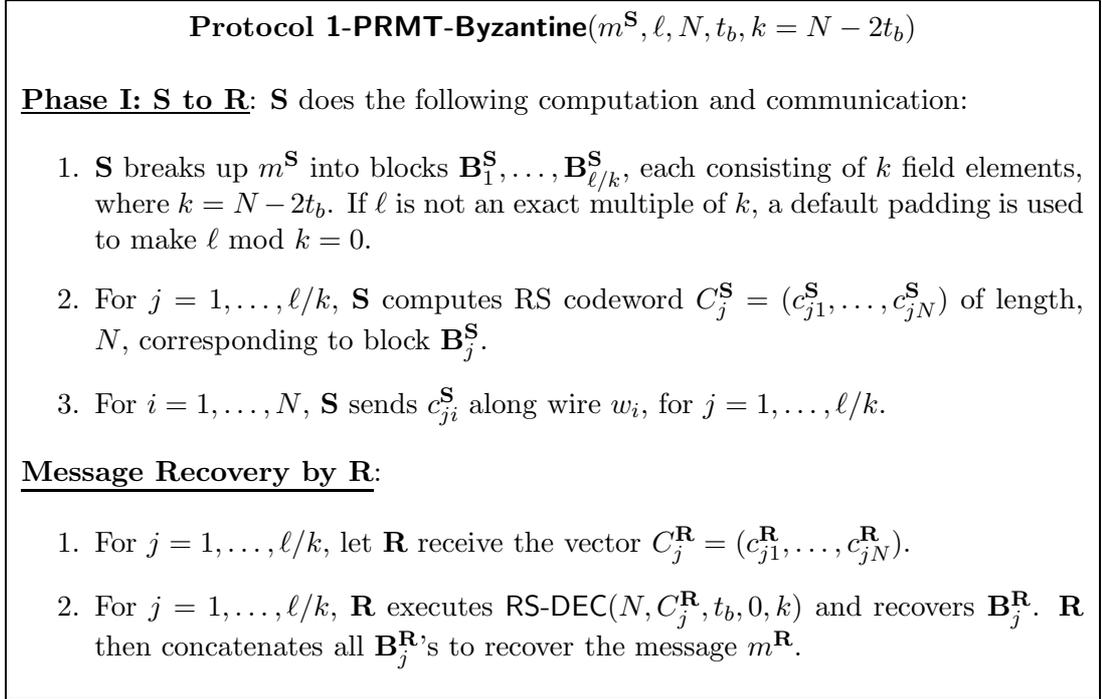
## 3.3 Protocols

Before presenting protocol 3-Optimal-PRMT-Static-Byzantine, we present few sub-protocols, which will be used as black-box in protocol 3-Optimal-PRMT-Static-Byzantine. We first present a single phase PRMT protocol called 1-PRMT-Byzantine, which is based on the properties of RS codes. The protocol will also be used in other subsequent chapters.

### 3.3.1 Single Phase PRMT Tolerating $\mathcal{A}_{t_b}^{static}$

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $N \geq 2t_b + 1$  wires,  $w_1, \dots, w_N$ , of which at most  $t_b$  wires can be under the control of  $\mathcal{A}_{t_b}^{static}$ . We design a single phase PRMT protocol called 1-PRMT-Byzantine, which allows  $\mathbf{S}$  to *reliably* send a message  $m^{\mathbf{S}}$  containing  $\ell \geq 1$  field

Figure 3.2: Single Phase Reliable Message Transmission Tolerating  $\mathcal{A}_{t_b}^{static}$



elements to **R**. The protocol is based on the properties of RS codes and is presented in Fig. 3.2.

We now prove the properties of protocol 1-PRMT-Byzantine.

**Lemma 3.6 (Correctness)** *Protocol 1-PRMT-Byzantine correctly delivers  $m^{\mathbf{S}}$ .*

PROOF: In order to show that **R** will correctly receive  $m^{\mathbf{S}}$ , we show that **R** will recover each  $\mathbf{B}_j^{\mathbf{S}}$  correctly. In the protocol, each  $\mathbf{B}_j^{\mathbf{S}}$  is of size  $k = N - 2t_b$  and is RS encoded into a codeword of length  $N$ , where  $N \geq 2t_b + 1$ . Corresponding to each  $\mathbf{B}_j^{\mathbf{S}}$ , **R** receives a vector  $C_j^{\mathbf{R}}$  of size  $N$  and this vector differs from the original codeword  $C_j^{\mathbf{S}}$  in at most  $t_b$  locations. So by putting  $N' = N \geq 2t_b + 1, k = N - 2t_b, c = t_b$  and  $d = 0$  in the inequality of Theorem 2.19, we find that **R** will be able to correct all the  $t_b$  errors in  $C_j^{\mathbf{R}}$  by applying RS-DEC to  $C_j^{\mathbf{R}}$ . Thus **R** correctly recovers  $\mathbf{B}_j^{\mathbf{S}}$ .  $\square$

**Lemma 3.7 (Communication Complexity)** *The communication complexity of protocol 1-PRMT-Byzantine is  $\mathcal{O}\left(\frac{N\ell}{N-2t_b}\right)$ .*

PROOF: Corresponding to each block of size  $k$ , **S** sends an RS codeword of length  $N$ . So communication complexity of the protocol is  $\mathcal{O}\left(\frac{N\ell}{k}\right) = \mathcal{O}\left(\frac{N\ell}{N-2t_b}\right)$ .  $\square$

### 3.3.2 Increasing the Throughput of Protocol 1-PRMT-Byzantine

Protocol 1-PRMT-Byzantine has another important property. *Consider the following scenario: Let **S** know that **R** has the knowledge of the exact identity of  $\alpha \leq t_b$  wires that are Byzantine corrupted. However **S** does not know the exact identity of those  $\alpha$  wires. If this is the case, then the following theorem holds:*

**Theorem 3.8 (Increased Throughput in 1-PRMT-Byzantine)** *Suppose  $\mathbf{S}$  knows that  $\mathbf{R}$  has the knowledge of the exact identity of  $\alpha \leq t_b$  wires that are Byzantine corrupted. Then in protocol 1-PRMT-Byzantine,  $\mathbf{S}$  can reliably send  $m^{\mathbf{S}}$  using block size  $k = (N - 2t_b) + \alpha$ . That is,  $\mathbf{S}$  can send  $\alpha$  field elements, in addition to what is allowed by Singleton bound. Moreover, the communication complexity of the protocol will be  $\mathcal{O}\left(\frac{N\ell}{(N-2t_b)+\alpha}\right)$*

*Proof:* Since  $\mathbf{R}$  is aware of the exact identity of  $\alpha$  Byzantine corrupted wires,  $\mathbf{R}$  can simply ignore the values received over these wires. So the length of each received vector  $C_j^{\mathbf{R}}$  will be  $N'$ , where  $N' = N - \alpha$ . Moreover  $C_j^{\mathbf{R}}$  will now differ from the original codeword  $C_j^{\mathbf{S}}$  in at most  $t_b - \alpha$  locations. So by putting  $N' = N - \alpha, k = (N - 2t_b) + \alpha, c = t_b - \alpha$  and  $d = 0$  in the inequality of Theorem 2.19, we find that by applying RS-DEC to  $C_j^{\mathbf{R}}$ ,  $\mathbf{R}$  will be able to correct all the  $t_b - \alpha$  errors in  $C_j^{\mathbf{R}}$  and hence correctly recover  $\mathbf{B}_j^{\mathbf{R}}$ .

Since  $k = (N - 2t_b) + \alpha$ , the communication complexity of the protocol will be  $\mathcal{O}\left(\frac{N\ell}{(N-2t_b)+\alpha}\right)$ .  $\square$

### 3.3.3 Reliably Communicating a Set of Conflicts

Consider the following scenario:  $\mathbf{S}$  selects at random  $n = 2t_b + 1$  polynomials each of degree  $t_b$ , denoted by  $p_i^{\mathbf{S}}(x)$ , where  $1 \leq i \leq n$ . Next through wire  $w_i$ ,  $\mathbf{S}$  sends to  $\mathbf{R}$  the following:

1. Polynomial  $p_i^{\mathbf{S}}(x)$ <sup>2</sup> and
2. For  $j = 1, \dots, n$ , the value  $r_{ji}^{\mathbf{S}} = p_j^{\mathbf{S}}(i)$ .

Assume that  $\mathbf{R}$  receives the following over wire  $w_i$ :

1. Polynomial  $p_i^{\mathbf{R}}(x)$ ;
2. The values  $r_{ji}^{\mathbf{R}}$ , for  $j = 1, \dots, n$ .

Now  $\mathbf{R}$  tries to find as many faults as he can find that occurred in the previous phase and communicate all his findings reliably back to  $\mathbf{S}$ . Towards this,  $\mathbf{R}$  first constructs what is known as *conflict graph*  $H = (\mathcal{W}, E)$ , where  $\mathcal{W} = \{w_1, w_2, \dots, w_n\}$  and  $(w_i, w_j) \in E$  if  $r_{ij}^{\mathbf{R}} \neq p_i^{\mathbf{R}}(j)$  or  $r_{ji}^{\mathbf{R}} \neq p_j^{\mathbf{R}}(i)$ . Corresponding to each edge  $(w_i, w_j) \in E$ ,  $\mathbf{R}$  adds a six tuple  $(w_i, w_j, r_{ij}^{\mathbf{R}}, p_i^{\mathbf{R}}(j), r_{ji}^{\mathbf{R}}, p_j^{\mathbf{R}}(i))$  to a list *Fault*. A naive and straightforward way of reliably sending *Fault* to  $\mathbf{S}$  is to broadcast *Fault* over all the  $n$  wires. In the worst case there can be  $\mathcal{O}(n^2)$  edges in the conflict graph and hence  $\mathcal{O}(n^2)$  tuples in the list *Fault*. So broadcasting *Fault* will require a communication complexity of  $\mathcal{O}(n^3)$  field elements.

An interesting question here is can  $\mathbf{R}$  reliably send *Fault* to  $\mathbf{S}$  tolerating  $\mathcal{A}_{t_b}^{\text{static}}$ , with a communication complexity less than  $\mathcal{O}(n^3)$ ? The answer is yes and that is what is done in [77], where the authors have given a method to reliably communicate *Fault* with a complexity of  $\mathcal{O}(n^2)$ . We call this method as *Matching technique*, which we use as a black box in protocol 3-Optimal-PRMT-Static-Byzantine. The method is formally explained in Fig. 3.3.

The following lemma and theorems taken from [77] show that by doing the computation and communication given in Fig. 3.3,  $\mathbf{R}$  will be able to reliably send the list of  $\mathcal{O}(n^2)$  confictions by communicating  $\mathcal{O}(n^2)$  field elements.

<sup>2</sup>We assume that the polynomial is sent by sending its  $t_b + 1$  coefficients.

Figure 3.3: Reliably Sending a List of Confliction Tolerating  $\mathcal{A}_{t_b}^{static}$

**Matching Technique**

**Computation and Communication by R:**

1. **R** initializes his *fault-list*, denoted by  $L_{fault}^{\mathbf{R}}$ , to  $\emptyset$ . **R** then constructs an undirected graph  $H = (\mathcal{W}, E)$  where  $\mathcal{W} = \{w_1, \dots, w_n\}$  and the edge  $(w_i, w_j) \in E$  if  $r_{ij}^{\mathbf{R}} \neq p_i^{\mathbf{R}}(j)$  or  $r_{ji}^{\mathbf{R}} \neq p_j^{\mathbf{R}}(i)$ .
2. If the degree of node  $w_i$  in the graph  $H$  constructed above is greater than  $t_b$  then **R** adds  $w_i$  to  $L_{fault}^{\mathbf{R}}$ .
3. Let  $H' = (\mathcal{W}', E')$  be the induced subgraph of  $H$  on the vertex set  $\mathcal{W}' = (\mathcal{W} \setminus L_{fault}^{\mathbf{R}})$ . Next, **R** finds a maximal matching<sup>a</sup>  $M \subseteq E'$  of graph  $H'$ .
4. For each edge  $(w_i, w_j)$  in  $H$  that does not belong to  $M$ , **R** associates the six-tuple  $\{w_i, w_j, r_{ij}^{\mathbf{R}}, r_{ji}^{\mathbf{R}}, p_j^{\mathbf{R}}(i), p_i^{\mathbf{R}}(j)\}$ . Let  $\{a_1, a_2, \dots, a_N\}$  be the edges in  $H$  that are not in  $M$ . Replacing each of these edges with its associated six-tuple, **R** gets a set of  $6N$  field elements, denoted by  $X = \{X_1, X_2, \dots, X_{6N}\}$ .
5. **R** then broadcasts the following to **S**
  - (a) The set  $L_{fault}^{\mathbf{R}}$ ;
  - (b) For each edge  $(w_i, w_j) \in M$ , the six tuple  $(w_i, w_j, r_{ij}^{\mathbf{R}}, r_{ji}^{\mathbf{R}}, p_i^{\mathbf{R}}(j), p_j^{\mathbf{R}}(i))$ .
6. Finally **R** reliably sends the list  $X$  to **S** by executing 1-PRMT-Byzantine( $X, |X|, n, t_b, |M| + |L_{fault}^{\mathbf{R}}|$ ) with **increased throughput** (see section 3.3.2).

<sup>a</sup> A subset  $M$  of the edges of  $H$ , is called a *matching* in  $H$  if no two edges in  $M$  are adjacent. A matching  $M$  is called *maximal* if it is not a proper subset of any other matching in the graph.

**Theorem 3.9 ([77])** *Given an undirected graph  $H = (V, E)$ , with a maximum degree  $\Delta$  and a maximal matching  $M$ , the number of edges  $|E|$  is less than or equal to  $(2|M|^2 + |M|\Delta)$ .*

**Lemma 3.10 ([77])** ***S** is guaranteed to receive the set  $X$  correctly in Matching Technique.*

PROOF: First notice that if  $(w_i, w_j) \in H$ , then either  $w_i$  or  $w_j$  or both are corrupted. This is because an honest wire will never conflict another honest wire. Similarly, if a node  $w_i$  in  $H$  has degree more than  $t_b$  then it implies that  $w_i$  is corrupted. This is because an honest wire may conflict at most  $t_b$  corrupted wires. This implies that all the wires listed in  $L_{fault}^{\mathbf{R}}$  are indeed corrupted. Since **R** broadcasts  $L_{fault}^{\mathbf{R}}$  and  $M$ , **S** will correctly receive them and hence will identify at least  $|L_{fault}^{\mathbf{R}}| + |M|$  corrupted wires. So from Theorem 3.8, by substituting  $N = 2t_b + 1$  and  $\alpha = |L_{fault}^{\mathbf{R}}| + |M|$ , we find that **S** will be able to reliably receive  $X$  at the end of 1-PRMT-Byzantine.  $\square$

**Theorem 3.11 ([77])** *The overall communication complexity done by **R** in Matching Technique is  $\mathcal{O}(n^2)$ .*

PROOF: First notice that  $|M| = \mathcal{O}(t_b)$ . This is because there are at most  $t_b$  corrupted wires and no two honest wires conflict each other. Similarly,  $|L_{fault}^{\mathbf{R}}| = \mathcal{O}(t_b)$ . This is because each wire in  $L_{fault}^{\mathbf{R}}$  is indeed corrupted. Also notice that the maximum degree of any vertex in the conflict graph  $H$  is  $t_b$ . Hence from Theorem 3.9 we find that the maximum number of edges in conflict graph  $H$  will be  $\mathcal{O}(t_b^2) = \mathcal{O}(n^2)$ . This further implies that  $|X| = \mathcal{O}(n^2)$ . So from Theorem 3.8, the communication complexity of sending  $X$  by executing 1-PRMT-Byzantine with block size of  $\alpha = |L_{fault}^{\mathbf{R}}| + |M|$  is  $\mathcal{O}(n^2)$ .  $\square$

Thus, the entire conflict graph is sent in two parts; first a matching  $M$  and list  $L_{fault}^{\mathbf{R}}$  is broadcast, which requires a communication complexity of  $\mathcal{O}(n^2)$ . Then the rest of the edges of the conflict graph are sent by communicating  $\mathcal{O}(n^2)$  field elements by using 1-PRMT-Byzantine with increased throughput.

We are now ready to present our three phase communication optimal PRMT protocol 3-Optimal-PRMT-Static-Byzantine, which we do in the next section.

### 3.3.4 Protocol 3-Optimal-PRMT-Static-Byzantine

We now give the formal description of protocol 3-Optimal-PRMT-Static-Byzantine. Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + 1$  wires, of which at most  $t_b$  wires are under the control of  $\mathcal{A}_{t_b}^{static}$ . The high level idea of the protocol is as follows: let  $m^{\mathbf{S}}$  be the message containing  $(t_b + 1)^2 = \Theta(n^2)$  field elements. We denote  $m^{\mathbf{S}}$  by  $m^{\mathbf{S}} = \{m_{ij}^{\mathbf{S}} : i = 0, \dots, t_b \text{ and } j = 0, \dots, t_b\}$ . During first phase,  $\mathbf{S}$  constructs a bi-variate polynomial  $F^{\mathbf{S}}(x, y)$  of degree  $t_b$  in  $x$  and  $y$ , such that the elements of  $m^{\mathbf{S}}$  constitutes the coefficients of  $F^{\mathbf{S}}(x, y)$ . From  $F^{\mathbf{S}}(x, y)$ ,  $\mathbf{S}$  obtains  $n$  univariate polynomials  $p_1^{\mathbf{S}}(x), \dots, p_n^{\mathbf{S}}(x)$ , where  $p_i^{\mathbf{S}}(x) = F^{\mathbf{S}}(x, i)$ . Then  $\mathbf{S}$  sends to  $\mathbf{R}$  over wire  $w_i$  the polynomial  $p_i^{\mathbf{S}}(x)$  and the value of every polynomial  $p_j^{\mathbf{S}}(x)$  at  $x = i$ . On receiving the values from  $\mathbf{S}$ , the receiver  $\mathbf{R}$  construct the conflict graph and reliably sends the conflict graph during second phase using Matching Technique.  $\mathbf{S}$  reliably receives the conflict graph and after doing local comparison, identifies all corrupted wires which delivered incorrect polynomials to  $\mathbf{R}$  during first phase. Notice that there can be at most  $t_b$  such corrupted wires.  $\mathbf{S}$  then broadcasts the identity of those corrupted wires to  $\mathbf{R}$  during third phase.  $\mathbf{R}$  on receiving the identity of those wires, ignores the polynomials received over those wires during the first phase.  $\mathbf{R}$  will be then left with at least  $t_b + 1$  correct univariate polynomials  $p_i^{\mathbf{S}}(x)$ 's, using which  $\mathbf{R}$  recovers  $F^{\mathbf{S}}(x, y)$  and hence the message  $m^{\mathbf{S}}$ . The protocol is formally presented in Fig. 3.4.

We now prove the properties of protocol 3-Optimal-PRMT-Static-Byzantine.

**Claim 3.12** *In protocol 3-Optimal-PRMT-Static-Byzantine, if  $w_i$  delivers  $p_i^{\mathbf{R}}(x) \neq p_i^{\mathbf{S}}(x)$  during Phase I, then  $\mathbf{S}$  will detect this at the end of Phase II and will include  $w_i$  in  $L_{fault}^{\mathbf{S}}$ . Thus, identities of all the wires which delivered incorrect polynomial during Phase I will be known to  $\mathbf{S}$  at the end of Phase II.*

PROOF: Suppose  $w_i$  is corrupted and delivers  $p_i^{\mathbf{R}}(x) \neq p_i^{\mathbf{S}}(x)$  during Phase I. In the worst case, these two polynomials can intersect at most at  $t_b$  points, since both are of degree  $t_b$ . Since there are at least  $n - t_b = t_b + 1$  honest wires and it may happen that  $p_i^{\mathbf{R}}(x) = p_i^{\mathbf{S}}(x)$  at most at  $t_b$   $j$ 's corresponding to  $t_b$  honest wires, it implies that there exists at least one honest wire, say  $w_j$ , such that  $p_i^{\mathbf{R}}(j) \neq p_i^{\mathbf{S}}(j)$ . Thus  $w_i$  and  $w_j$  will conflict each other and so the edge  $(w_i, w_j)$  will be present in the conflict graph  $H$ . Now by the property of Matching Technique, corresponding to the edge  $(w_i, w_j)$  in  $H$ ,  $\mathbf{S}$  will reliably receive the six tuple  $(w_i, w_j, r_{ij}^{\mathbf{R}}, p_i^{\mathbf{R}}(j), r_{ji}^{\mathbf{R}}, p_j^{\mathbf{R}}(i))$ . So after doing

Figure 3.4: Three Phase Communication Optimal PRMT Tolerating  $\mathcal{A}_{t_b}^{static}$

<p><b>Protocol 3-Optimal-PRMT-Static-Byzantine</b> (<math>m^S = \{m_{ij}^S : i = 0, \dots, t_b \text{ and } j = 0, \dots, t_b\}</math>)</p> <p><b>Phase I: S to R</b></p> <ol style="list-style-type: none"> <li>1. <b>S</b> constructs a bivariate polynomial <math>F^S(x, y)</math> of degree <math>t_b</math> in <math>x</math> and <math>y</math> where <math>F^S(x, y) = \sum_{i=0}^{t_b} \sum_{j=0}^{t_b} m_{ij}^S x^i y^j</math>.</li> <li>2. For <math>i = 1, \dots, n</math>, <b>S</b> constructs univariate polynomial <math>p_i^S(x) = F^S(x, i)</math>.</li> <li>3. <b>S</b> sends the following to <b>R</b> over wire <math>w_i</math>, for <math>i = 1, \dots, n</math>: <ol style="list-style-type: none"> <li>(a) Polynomial <math>p_i^S(x)</math>;</li> <li>(b) Value <math>r_{ji}^S = p_j^S(i)</math>, for <math>j = 1, \dots, n</math>.</li> </ol> </li> </ol> <p><b>Phase II: R to S</b></p> <ol style="list-style-type: none"> <li>1. Let <b>R</b> receive the following over wire <math>w_i</math>, for <math>i = 1, \dots, n</math>: <ol style="list-style-type: none"> <li>(a) Polynomial <math>p_i^R(x)</math>;</li> <li>(b) Value <math>r_{ji}^R</math>, for <math>j = 1, \dots, n</math>.</li> </ol> </li> <li>2. <b>R</b> constructs the conflict graph <math>H</math> and reliably sends it to <b>S</b> using Matching Technique.</li> </ol> <p><b>Phase III: S to R</b></p> <ol style="list-style-type: none"> <li>1. <b>S</b> reliably receives the conflict graph <math>H</math>. Thus corresponding to each edge <math>(w_i, w_j)</math> in <math>H</math>, <b>S</b> reliably receives the six-tuple <math>(w_i, w_j, r_{ij}^R, p_i^R(j), r_{ji}^R, p_j^R(i))</math>.</li> <li>2. <b>S</b> initializes <math>L_{fault}^S = L_{fault}^R</math><sup>a</sup>. Then for every received six-tuple <math>(w_i, w_j, r_{ij}^R, p_i^R(j), r_{ji}^R, p_j^R(i))</math>, <b>S</b> does the following computation: <ol style="list-style-type: none"> <li>(a) <b>S</b> checks <math>r_{ij}^R \stackrel{?}{=} r_{ij}^S</math> or <math>p_i^R(j) \stackrel{?}{=} p_i^S(j)</math>. If the first test fails then <b>S</b> concludes that <math>w_j</math> is corrupted and adds <math>w_j</math> to <math>L_{fault}^S</math>. On the other hand if the second test fails then <b>S</b> concludes that <math>w_i</math> is corrupted and adds <math>w_i</math> to <math>L_{fault}^S</math>.</li> <li>(b) <b>S</b> checks <math>r_{ji}^R \stackrel{?}{=} r_{ji}^S</math> or <math>p_j^R(i) \stackrel{?}{=} p_j^S(i)</math>. If the first test fails then <b>S</b> concludes that <math>w_i</math> is corrupted and adds <math>w_i</math> to <math>L_{fault}^S</math>. On the other hand if the second test fails then <b>S</b> concludes that <math>w_j</math> is corrupted and adds <math>w_j</math> to <math>L_{fault}^S</math>.</li> </ol> </li> <li>3. <b>S</b> broadcasts <math>L_{fault}^S</math> to <b>R</b> and terminates the protocol.</li> </ol> <p><b>Message Recovery by R</b></p> <ol style="list-style-type: none"> <li>1. <b>R</b> correctly receives <math>L_{fault}^S</math> and identifies all corrupted <math>w_i</math>'s which delivered incorrect <math>p_i^S(x)</math>'s during <b>Phase I</b>.</li> <li>2. <b>R</b> neglects the polynomials received over the wires in <math>L_{fault}^S</math> during <b>Phase I</b>.</li> <li>3. Using the remaining polynomials, <b>R</b> reconstructs <math>F^S(x, y)</math> and hence the message <math>m^S</math> and terminates the protocol.</li> </ol>
--

<sup>a</sup> Recall that during Matching Technique, **S** receives  $L_{fault}^R$  from **R**.

local computation, **S** will find that  $p_i^R(j) \neq p_i^S(j)$  and hence will conclude that  $w_i$  has delivered  $p_i^R(x) \neq p_i^S(x)$ . Thus **S** will include  $w_i$  in  $L_{fault}^S$ .  $\square$

**Lemma 3.13 (Correctness)** *In protocol 3-Optimal-PRMT-Static-Byzantine, **R** will be able to correctly recover  $m^S$  at the end of Phase III.*

PROOF: From Claim 3.12, at the end of **Phase II**, **S** will identify all corrupted  $w_i$  who has delivered incorrect  $p_i^R(x) \neq p_i^S(x)$  to **R** during **Phase I** and will include such  $w_i$ 's in  $L_{fault}^S$ . Since **S** broadcasts  $L_{fault}^S$  during **Phase III**, **R** will also come to know the identity of such  $w_i$ 's at the end of **Phase III** and hence will neglect them. Now

notice that there can be at most  $t_b$  such corrupted  $w_i$ 's. Hence at the end of **Phase III**, **R** will come to know about all the  $p_i^{\mathbf{S}}(x)$ 's (at least  $t_b + 1$ ) which he has received correctly during **Phase I**. Now it is easy to see that using those  $p_i^{\mathbf{S}}(x)$ 's, **R** will be able to correctly reconstruct  $F^{\mathbf{S}}(x, y)$ . This is because  $F^{\mathbf{S}}(x, y)$  is a bivariate polynomial of degree  $t_b$  in  $x$  and  $y$  and hence it can be reconstructed using  $t_b + 1$  correct  $p_i^{\mathbf{S}}(x)$ 's. Since all corrupted  $p_i^{\mathbf{R}}(x)$ 's will be eliminated by **R**, all polynomials  $p_i^{\mathbf{S}}(x)$ , where  $w_i \in \{w_1, \dots, w_n\} \setminus L_{fault}^{\mathbf{S}}$  are correct. **R** can use any  $t_b + 1$  of these polynomials to reconstruct  $F^{\mathbf{S}}(x, y)$ . Once  $F^{\mathbf{S}}(x, y)$  is reconstructed, **R** will get  $m^{\mathbf{S}}$ , as the coefficients of  $F^{\mathbf{S}}(x, y)$  are the elements of  $m^{\mathbf{S}}$ .  $\square$

**Lemma 3.14 (Communication Complexity)** *The total communication complexity of protocol 3-Optimal-PRMT-Static-Byzantine is  $\mathcal{O}(n^2)$ .*

PROOF: During **Phase I**, **S** sends a polynomial of degree  $t_b$  and  $n$  values over each wire. This will incur a communication complexity of  $\mathcal{O}(n^2)$ . During **Phase II**, **R** reliably sends the conflict graph using Matching Technique, which from Theorem 3.11 requires a communication complexity of  $\mathcal{O}(n^2)$ . During **Phase III**, **S** broadcasts  $L_{fault}^{\mathbf{S}}$ . Since  $|L_{fault}^{\mathbf{S}}| \leq t_b$ , **Phase III** will incur a communication cost of  $\mathcal{O}(n^2)$ . Thus the overall communication complexity of protocol 3-Optimal-PRMT-Static-Byzantine is  $\mathcal{O}(n^2)$ .  $\square$

**Theorem 3.15** *Protocol 3-Optimal-PRMT-Static-Byzantine is a communication optimal PRMT protocol which achieves reliability with constant factor overhead.*

PROOF: From Theorem 3.4, any three phase PRMT protocol over  $n = 2t_b + 1$  wires must communicate  $\Omega\left(\frac{n\ell}{n-t_b}\right) = \Omega(\ell)$  field elements to reliably send a message containing  $\ell$  field elements against  $\mathcal{A}_{t_b}^{static}$ . This is because  $n = 2t_b + 1$  and hence  $n - t_b = \Theta(n)$ . Now substituting  $\ell = (t_b + 1)^2$ , we find that any three phase PRMT protocol over  $n = 2t_b + 1$  wires has to communicate  $\Omega(t_b^2) = \Omega(n^2)$  field elements. From Lemma 3.14, the total communication complexity of protocol 3-Optimal-PRMT-Static-Byzantine is  $\mathcal{O}(n^2)$ . Hence the protocol is communication optimal. Moreover, it is easy to see that the protocol achieves reliability with constant factor overhead.  $\square$

**Theorem 3.16** *Let  $\mathcal{N}$  be an undirected synchronous network, under the influence of  $\mathcal{A}_{t_b}^{static}$ , where **S** and **R** are connected by  $n = 2t_b + 1$  wires. Then three phases are necessary and sufficient for the existence of any PRMT protocol which achieves reliability with constant factor overhead.*

PROOF: Follows from Theorem 3.4 and Theorem 3.15.  $\square$

### 3.4 Concluding Remarks and Open Problems

In this chapter, we presented a three phase communication optimal PRMT protocol in undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{static}$ . Moreover, we have shown that our protocol is phase optimal, as well as require minimum connectivity. This, along with Theorem 3.2 completely settles the issue of POSSIBILITY, FEASIBILITY and OPTIMALITY of PRMT in undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{static}$ . These results are summarized in Fig. 3.5.

From Fig. 3.5, we find that protocol 3-Optimal-PRMT-Static-Byzantine is communication optimal only if the message contains  $\ell = \Theta(n^2)$  field elements. This leads to our first open problem which is as follows:

Figure 3.5: Summary of the Results for PRMT in Undirected Synchronous Network Tolerating  $\mathcal{A}_{t_b}^{static}$

Number of Phases ( $r$ )	Connectivity Requirement ( $n$ )	Lower Bound on Communication Complexity	Upper Bound
$r \leq 2$	$n \geq 2t_b + 1$ [28]	$\Omega\left(\frac{n\ell}{n-2t_b}\right)$ [81]	Broadcast protocol: $n = 2t_b + 1$ , Communication complexity = $\mathcal{O}(n\ell)$
$r \geq 3$	$n \geq 2t_b + 1$ [28]	$\Omega\left(\frac{n\ell}{n-t_b}\right)$ [81]	Protocol 3-Optimal-PRMT-Static-Byzantine: $n = 2t_b + 1, \ell = \Theta(n^2)$ Communication complexity = $\mathcal{O}(\ell)$

**Open Problem 1** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + 1$  wires. Then does there exist a multiphase (more than two phase) PRMT protocol which reliably sends a message containing  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements, tolerating  $\mathcal{A}_{t_b}^{static}$ , for all values of  $\ell$  ?*

## Chapter 4

# PRMT in Undirected Networks Tolerating Mobile Byzantine Adversary

In this chapter, we study PRMT in *undirected synchronous network, tolerating threshold mobile Byzantine adversary*. PRMT in the presence of static adversary has received quiet a bit of attention in the past [28, 49, 81]. However, as first noticed by Ostrovsky et al. [50], the static model implicitly assumes that the number of dishonest nodes in the network is independent of the execution time of the protocol. This is usually not true in practice. Furthermore, since a corrupted node could be corrected given sufficient time, Ostrovsky et al. [50] proposed the *mobile* adversary model wherein the adversary could move around the network whilst still corrupting up to  $t_b$  nodes at any given instant. Subsequently, extensive research efforts on tolerating mobile adversaries have resulted in what is now well-known as *proactive* security [39, 32, 38, 7]. However, in the context of PRMT, not too much is known regarding the mobility of the adversary. So in this chapter, we completely settle the issue of POSSIBILITY, FEASIBILITY and OPTIMALITY of PRMT in *undirected synchronous network tolerating threshold mobile Byzantine adversary*. We begin with the network model used in this chapter.

### 4.1 Underlying Network Model and Adversary Settings

The network model is similar to the one used in the previous chapter. However, instead of threshold static adversary  $\mathcal{A}_{t_b}^{static}$ , we assume the presence of a threshold mobile adversary  $\mathcal{A}_{t_b}^{mobile}$ . Unlike  $\mathcal{A}_{t_b}^{static}$ , who controls the *same* set of  $t_b$  wires throughout the protocol,  $\mathcal{A}_{t_b}^{mobile}$  may control *different* set of  $t_b$  wires during different phases of the protocol. Thus if some wire  $w_j$  is under the control of  $\mathcal{A}_{t_b}^{mobile}$  in  $i^{th}$  phase of a protocol, then it does not imply that  $w_j$  will be corrupted in  $(i+1)^{th}$  phase also, unless  $\mathcal{A}_{t_b}^{mobile}$  controls  $w_j$  in  $(i+1)^{th}$  phase also. Moreover, by controlling  $w_j$  in  $i^{th}$  phase, the adversary  $\mathcal{A}_{t_b}^{mobile}$  will not get any information about the communication done over wire  $w_j$  in earlier phase(s), if  $\mathcal{A}_{t_b}^{mobile}$  has not controlled  $w_j$  in earlier phase(s). This is because we assume that every intermediate node along a wire immediately erases all information from its local memory at the end of a phase.

We next summarize the existing literature and our result for PRMT in undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{mobile}$ .

## 4.2 Existing Literature

PRMT problem in the presence of  $\mathcal{A}_{t_b}^{mobile}$  was studied in [82], who gave the following characterization:

**Theorem 4.1 ([82])** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n$  wires and let  $r \geq 1$ . Then any  $r$ -phase PRMT protocol tolerating  $\mathcal{A}_{t_b}^{mobile}$  is possible iff  $n \geq 2t_b + 1$ .*

The above theorem is quite interesting because it shows that the connectivity requirement for PRMT is the same against both  $\mathcal{A}_{t_b}^{static}$ , as well as  $\mathcal{A}_{t_b}^{mobile}$ . This is non-intuitive because  $\mathcal{A}_{t_b}^{mobile}$  is more powerful than  $\mathcal{A}_{t_b}^{static}$  and hence it is expected that the connectivity requirement for PRMT against  $\mathcal{A}_{t_b}^{mobile}$  should be more than the network connectivity required for  $\mathcal{A}_{t_b}^{static}$ . Indeed, if  $n = 2t_b + 1$ , then  $\mathbf{S}$  can reliably send a message by simply broadcasting it over all the  $n$  wires. Then irrespective of whether the adversary is static or mobile,  $\mathbf{R}$  will correctly receive the message.

Since  $\mathcal{A}_{t_b}^{mobile}$  is more powerful than  $\mathcal{A}_{t_b}^{static}$ , the lower bound on the communication complexity of PRMT against  $\mathcal{A}_{t_b}^{static}$ , as given in Theorem 3.3 and Theorem 3.4 must hold against  $\mathcal{A}_{t_b}^{mobile}$ . Thus we have the following theorems:

**Theorem 4.2** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + 1$  wires and let  $r \leq 2$ . Then any  $r$ -phase PRMT protocol tolerating  $\mathcal{A}_{t_b}^{mobile}$  must communicate  $\Omega\left(\frac{n\ell}{n-2t_b}\right)$  field elements to reliably send a message containing  $\ell$  field elements.*

**Theorem 4.3** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + 1$  wires and let  $r \geq 3$ . Then any  $r$ -phase PRMT protocol tolerating  $\mathcal{A}_{t_b}^{mobile}$  must communicate  $\Omega\left(\frac{n\ell}{n-t_b}\right)$  field elements to reliably send a message containing  $\ell$  field elements.*

It is easy to see that the lower bound given in Theorem 4.2 is tight. Specifically, let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + 1$  wires, which is the minimum number of wires required for any PRMT protocol against  $\mathcal{A}_{t_b}^{mobile}$ . Then  $\mathbf{S}$  can reliably send a message  $m^{\mathbf{S}}$  containing  $\ell$  field elements by simply broadcasting it. This will take one phase and will require a communication complexity of  $\mathcal{O}\left(\frac{n\ell}{n-2t_b}\right) = \mathcal{O}(n\ell)$  field elements.

If  $n = 2t_b + 1$ , then  $n = \Theta(t_b)$  and hence from Theorem 4.3, any three or more phase PRMT protocol must communicate  $\Omega\left(\frac{n\ell}{n-t_b}\right) = \Omega(\ell)$  field elements to reliably send a message containing  $\ell$  field elements against  $\mathcal{A}_{t_b}^{mobile}$ . To the best of our knowledge, there is no PRMT protocol tolerating  $\mathcal{A}_{t_b}^{mobile}$ , which satisfies this bound. This motivates us to design communication optimal PRMT protocol tolerating  $\mathcal{A}_{t_b}^{mobile}$ .

### 4.2.1 Our Contribution

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + 1$  wires. We then design a three phase communication optimal PRMT protocol called 3-Optimal-PRMT-Mobile-Byzantine, tolerating  $\mathcal{A}_{t_b}^{mobile}$ , which reliably sends a message containing  $\ell = \Theta(n^3)$  field elements by communicating  $\mathcal{O}(n^3)$  field elements. From Theorem 4.3, our protocol is *phase optimal*. Moreover, from Theorem 4.2, our protocol requires minimum connectivity. The only demerit of our protocol is that it achieves optimality only if  $\ell = \Theta(n^3)$ .

In the previous chapter, we presented a three phase PRMT protocol called 3-Optimal-PRMT-Static-Byzantine, which reliably sends a message containing  $\ell = \Theta(n^2)$  field elements by communicating  $\mathcal{O}(n^2)$  field elements, tolerating  $\mathcal{A}_{t_b}^{static}$ . An interesting question here is why protocol 3-Optimal-PRMT-Static-Byzantine will not work against  $\mathcal{A}_{t_b}^{mobile}$ . We answer this question in the next section.

### 4.3 Limitations of Matching Technique

The main reason why protocol 3-Optimal-PRMT-Static-Byzantine will not work against  $\mathcal{A}_{t_b}^{mobile}$  is that protocol 3-Optimal-PRMT-Static-Byzantine uses Matching Technique to reliably send the conflict list during **Phase II**. The Matching Technique will be able to reliably send the conflict list against  $\mathcal{A}_{t_b}^{static}$ . However, it will fail to reliably send the conflict list against  $\mathcal{A}_{t_b}^{mobile}$ . This is because  $\mathcal{A}_{t_b}^{mobile}$  can corrupt different set of  $t_b$  wires in each phase. In Matching Technique, **R** first broadcasts a maximal matching  $M$  and the list  $L_{fault}^{\mathbf{R}}$ , which facilitates **S** to identify  $|M| + |L_{fault}^{\mathbf{R}}|$  wires which were corrupted during **Phase I**. If the adversary is static, then the same set of  $|M| + |L_{fault}^{\mathbf{R}}|$  wires will be corrupted in all the three phases and hence **S** can neglect them. **S** can recover the remaining portion of the conflict graph by the properties of 1-PRMT-Byzantine with **increased throughput** (see Theorem 3.8).

However, if the adversary is mobile, then the above logic will not work. This is because the set of  $|M| + |L_{fault}^{\mathbf{R}}|$  wires which were corrupted in **Phase I** may not be corrupted in remaining phases of the protocol. In the worst case, these  $|M| + |L_{fault}^{\mathbf{R}}|$  wires may not be under the control of the adversary at all during second phase. So if **S** neglects these wires, then **S** will lose  $|M| + |L_{fault}^{\mathbf{R}}|$  correct values and hence 1-PRMT-Byzantine (with **increased throughput**) will fail to correctly deliver the remaining portion of the conflict graph. So against a mobile adversary, the only way of reliably sending the conflict graph seems to broadcast it. But this will require a communication complexity of  $\mathcal{O}(n^3)$  field elements. To circumvent this situation, we present a technique called Union Technique, which will be used in our protocol 3-Optimal-PRMT-Mobile-Byzantine.

### 4.4 Protocols

We first begin with the description of Union Technique, which will be used in protocol 3-Optimal-PRMT-Mobile-Byzantine.

#### 4.4.1 Union Technique

Now recall the same scenario described in Section 3.3.3. **S** and **R** are connected by  $n = 2t_b + 1$  wires. **S** selects at random  $n$  polynomials each of degree  $t_b$ , denoted by  $p_i^{\mathbf{S}}(x)$ , where  $1 \leq i \leq n$ . Next through wire  $w_i$ , **S** sends to **R** the following:

1. Polynomial  $p_i^{\mathbf{S}}(x)$  and
2. For  $j = 1, \dots, n$ , the value  $r_{ji}^{\mathbf{S}} = p_j^{\mathbf{S}}(i)$ .

Assume that **R** receives the following over wire  $w_i$ :

1. Polynomial  $p_i^{\mathbf{R}}(x)$ ;
2. The values  $r_{ji}^{\mathbf{R}}$ , for  $j = 1, \dots, n$ .

Let  $B^{\mathbf{R}}$  denote the set of  $n$  polynomials and  $n^2$  values as received by **R**. Using  $B^{\mathbf{R}}$ , **R** can construct a conflict graph. Now, in our three phase PRMT protocol 3-Optimal-PRMT-Mobile-Byzantine, during **Phase I**, instead of a single set  $B^{\mathbf{R}}$ , **R** receives  $n$  such sets denoted as  $B_k^{\mathbf{R}}$ , for  $k = 1, \dots, n$ , where each  $B_k^{\mathbf{R}}$  contains  $n$  polynomials, denoted by  $p_{k,i}^{\mathbf{R}}(x)$ ,  $i = 1, \dots, n$  and  $n^2$  values, denoted by  $r_{k,i,j}^{\mathbf{R}}$ ,  $1 \leq i, j \leq n$ . The flow of

Figure 4.1: Data Flow Over the  $n$  Wires During **Phase I** of Protocol 3-Optimal-PRMT-Mobile-Byzantine

Wire	$B_1^{\mathbf{R}}$	...	$B_n^{\mathbf{R}}$
$w_1$	$p_{1,1}^{\mathbf{R}}(x) r_{1,1,1}^{\mathbf{R}}, \dots, r_{1,n,1}^{\mathbf{R}}$	...	$p_{n,1}^{\mathbf{R}}(x) r_{n,1,1}^{\mathbf{R}}, \dots, r_{n,n,1}^{\mathbf{R}}$
...	...	...	...
$w_i$	$p_{1,i}^{\mathbf{R}}(x) r_{1,1,i}^{\mathbf{R}}, \dots, r_{1,n,i}^{\mathbf{R}}$	...	$p_{n,i}^{\mathbf{R}}(x) r_{n,1,i}^{\mathbf{R}}, \dots, r_{n,n,i}^{\mathbf{R}}$
...	...	...	...
$w_n$	$p_{1,n}^{\mathbf{R}}(x) r_{1,1,n}^{\mathbf{R}}, \dots, r_{1,n,n}^{\mathbf{R}}$	...	$p_{n,n}^{\mathbf{R}}(x) r_{n,1,n}^{\mathbf{R}}, \dots, r_{n,n,n}^{\mathbf{R}}$

information over  $n$  wires during **Phase I** of 3-Optimal-PRMT-Mobile-Byzantine is given in Fig. 4.1.

$\mathbf{R}$  then constructs conflict graph  $H_k$  using the set  $B_k^{\mathbf{R}}$ . For each  $H_k$ , we can say the following from the proof of Claim 3.12: if during **Phase I**,  $\mathbf{R}$  receives a corrupted polynomial  $p_{k,i}^{\mathbf{R}}(x) \neq p_{k,i}^{\mathbf{S}}(x)$  over  $w_i$ , then there exist at least one edge  $(w_i, w_j)$  in  $H_k$ , where  $w_j$  is an honest wire. If  $\mathbf{R}$  broadcasts all the  $n$  conflict graphs to  $\mathbf{S}$ , then  $\mathbf{S}$  will correctly receive  $H_k$  (even if the adversary is mobile) and hence from Claim 3.12, after doing local comparison,  $\mathbf{S}$  can find out the identity of all faulty wires  $w_i$  over which  $\mathbf{R}$  has received at least one faulty  $p_{k,i}^{\mathbf{R}}(x)$  during **Phase I**, where  $k \in \{1, \dots, n\}$ . However, broadcasting all conflicting graphs requires communicating  $\mathcal{O}(n^4)$  field elements, as each conflict graph may have  $\mathcal{O}(n^2)$  edges. So we now introduce a method of combining  $n$  conflict graphs into a single conflict graph  $H^{\mathbf{R}}$ . By broadcasting  $H^{\mathbf{R}}$  to  $\mathbf{S}$ , the receiver  $\mathbf{R}$  can ensure that  $\mathbf{S}$  will be able to identify all  $w_i$ 's over which  $\mathbf{R}$  has received at least one faulty polynomial  $p_{k,i}^{\mathbf{R}}(x)$ . However, broadcasting  $H^{\mathbf{R}}$  will require a communication complexity of  $\mathcal{O}(n^3)$ .

The combined conflict graph  $H^{\mathbf{R}} = (V, E)$  will have vertices and edges as follows:  $V = \{w_1, w_2, \dots, w_n\}$  and  $E = \{(w_i, w_j)\}$  where edge  $(w_i, w_j) \in E$  if the edge  $(w_i, w_j)$  occurs in at least one  $H_k, k = 1, \dots, n$ . Since an edge  $(w_i, w_j)$  can occur in multiple  $H_k$ 's,  $\mathbf{R}$  considers  $(w_i, w_j)$  from the minimum indexed  $H_\gamma$  among all such  $H_k$ 's, keeping a note that  $(w_i, w_j)$  is added from  $H_\gamma$ . For each  $(w_i, w_j) \in E$ ,  $\mathbf{R}$  adds a seven tuple  $\{w_i, w_j, \gamma, p_{\gamma,i}^{\mathbf{R}}(j), r_{\gamma,i,j}^{\mathbf{R}}, p_{\gamma,j}^{\mathbf{R}}(i), r_{\gamma,j,i}^{\mathbf{R}}\}$  to a list  $X$ , provided  $(w_i, w_j)$  is taken from  $H_\gamma$ . This indicates that in the set  $B_\gamma^{\mathbf{R}}$ , the wires  $w_i$  and  $w_j$  conflict with each other. It is easy to see that there can be  $\mathcal{O}(n^2)$  edges in  $H^{\mathbf{R}}$  and hence  $\mathcal{O}(n^2)$  tuples in  $X$ . We call this method of generating a single conflict list from  $n$  conflict lists as **Union Technique**.

In the next theorem, we prove that  $\mathbf{S}$  can identify all faulty wires over which  $\mathbf{R}$  received at least one faulty polynomial after correctly receiving  $X$ .

**Theorem 4.4** *In the Union Technique, if  $\mathbf{R}$  broadcasts  $X$  to  $\mathbf{S}$ , then  $\mathbf{S}$  identifies all faulty wires  $w_i$  over which  $\mathbf{R}$  has received at least one corrupted polynomial  $p_{k,i}^{\mathbf{R}}(x) \neq p_{k,i}^{\mathbf{S}}(x)$  during **Phase I**.*

PROOF: Suppose during **Phase I**,  $\mathbf{R}$  receives a faulty polynomial  $p_{k,i}^{\mathbf{R}}(x) \neq p_{k,i}^{\mathbf{S}}(x)$  over  $w_i$ . Then from Claim 3.12, there exists at least one edge  $(w_i, w_j) \in H_k$ , where  $w_j$  is an honest wire. Since the combined conflict graph  $H^{\mathbf{R}}$  is formed by considering all the edges in the individual  $H_k$ 's,  $1 \leq k \leq n$ , list  $X$  must have a seven tuple  $\{w_i, w_j, \gamma, p_{\gamma,i}^{\mathbf{R}}(j), r_{\gamma,i,j}^{\mathbf{R}}, p_{\gamma,j}^{\mathbf{R}}(i), r_{\gamma,j,i}^{\mathbf{R}}\}$ . Now there are following two possibilities:

1.  $\gamma = k$ : This indicates that the seven tuple exactly corresponds to the edge  $(w_i, w_j) \in H_k$ .

2.  $\gamma < k$ : This indicates that the seven tuple corresponds to the edge  $(w_i, w_j) \in H_\gamma$ , which implies that wire  $w_i$  has either delivered incorrect  $p_{\gamma,i}^{\mathbf{R}}(x) \neq p_{\gamma,i}^{\mathbf{S}}(x)$  or an incorrect value  $r_{\gamma,j,i}^{\mathbf{R}} \neq r_{\gamma,j,i}^{\mathbf{S}}$ . However, what ever may be the case, adding the seven tuple for the edge  $(w_i, w_j) \in H_\gamma$  to the list  $X$  will not affect in identifying  $w_i$  as as a corrupted wire.

Thus, for each faulty  $w_i$  delivering at least one incorrect polynomial during **Phase I**, there exists a seven tuple in  $X$ . If  $\mathbf{R}$  broadcasts  $X$ , then  $\mathbf{S}$  will correctly receive it (even if the adversary is mobile) and hence after performing local verification,  $\mathbf{S}$  will identify all faulty wires, over which  $\mathbf{R}$  received at least one faulty  $p_{k,i}^{\mathbf{R}}(x)$   $\square$

We are now well equipped to present protocol 3-Optimal-PRMT-Mobile-Byzantine, which we present in the next section.

#### 4.4.2 Protocol 3-Optimal-PRMT-Mobile-Byzantine

The protocol is similar to protocol 3-Optimal-PRMT-Static-Byzantine. However, instead of using Matching Technique, we use Union Technique to reliably send the conflict list.

Intuitively, protocol 3-Optimal-PRMT-Mobile-Byzantine works as follows:  $\mathbf{S}$  selects  $n$  bivariate polynomials, each of degree  $t_b$  in  $x$  and  $y$ , whose coefficients are the elements of the message to be sent.  $\mathbf{S}$  then generates  $n$  sets  $B_k^{\mathbf{S}}, k = 1, \dots, n$  from the  $n$  bivariate polynomials and communicates them to  $\mathbf{R}$  during **Phase I**. On receiving  $n B_k^{\mathbf{R}}$ 's,  $\mathbf{R}$  first constructs  $n$  conflict graphs  $H_k$ 's and then combine all of them to a single graph  $H^{\mathbf{R}}$ , using Union Technique.  $\mathbf{R}$  then broadcasts to  $\mathbf{S}$  the list of seven tuples corresponding to the conflict graph  $H^{\mathbf{R}}$  during **Phase II**. In **Phase III**,  $\mathbf{S}$  identifies all faulty wires which delivered incorrect polynomials to  $\mathbf{R}$  during **Phase I** and broadcasts their identities to  $\mathbf{R}$ . Finally,  $\mathbf{R}$  recovers the message by reconstructing all the  $n$  bivariate polynomials by ignoring the faulty wires, which delivered incorrect polynomials during **Phase I**. The protocol is formally given in Fig. 4.2.

We now prove the the properties of protocol 3-Optimal-PRMT-Mobile-Byzantine.

**Lemma 4.5 (Correctness)** *In protocol 3-Optimal-PRMT-Mobile-Byzantine,  $\mathbf{R}$  will always be able to correctly recover the message.*

PROOF: In protocol 3-Optimal-PRMT-Mobile-Byzantine, to recover  $m^{\mathbf{S}}$ ,  $\mathbf{R}$  should be able to interpolate each bivariate polynomial  $F_k^{\mathbf{S}}(x, y)$ , for  $k = 1, \dots, n$ . Since each  $F_k^{\mathbf{S}}(x, y)$  is of degree  $t_b$  in both  $x$  and  $y$ ,  $\mathbf{R}$  requires  $t_b + 1$  correct  $F_k^{\mathbf{S}}(x, i) = p_{k,i}^{\mathbf{S}}(x)$ 's to recover  $F_k^{\mathbf{S}}(x, y)$ . Since among the  $n$  wires, at most  $t_b$  can be corrupted during **Phase I**,  $\mathbf{R}$  will receive at least  $t_b + 1$  correct  $p_{k,i}^{\mathbf{S}}(x)$ 's. During **Phase II**,  $\mathbf{R}$  constructs  $n$  conflict graph  $H_k, 1 \leq k \leq n$  and combine them into a single conflict graph  $H^{\mathbf{R}}$  using Union Technique, forms the corresponding list of seven tuples  $X$  and broadcasts it to  $\mathbf{S}$ . So  $\mathbf{S}$  will correctly receive  $X$ . From Theorem 4.4, on receiving  $X$ ,  $\mathbf{S}$  identifies all faulty wires over which  $\mathbf{R}$  has received at least one faulty polynomial during **Phase I** and adds them to  $L_{fault}$  and broadcasts  $L_{fault}$  to  $\mathbf{R}$ . So  $\mathbf{R}$  correctly receives  $L_{fault}$  and neglects all the  $n$  polynomials received over each  $w_i \in L_{fault}$ . Since  $|L_{fault}| \leq t_b$ ,  $\mathbf{R}$  will be left with at least  $t_b + 1$  correct  $p_{k,i}^{\mathbf{S}}(x)$ 's, for each  $1 \leq k \leq n$ , using which  $\mathbf{R}$  recovers each  $F_k^{\mathbf{S}}(x, y)$  and hence  $m^{\mathbf{S}}$ .  $\square$

**Lemma 4.6 (Communication Complexity)** *The total communication complexity of protocol 3-Optimal-PRMT-Mobile-Byzantine is  $\mathcal{O}(n^3)$ .*

Figure 4.2: Three Phase Communication Optimal PRMT Tolerating  $\mathcal{A}_{t_b}^{mobile}$

**Protocol 3-Optimal-PRMT-Mobile-Byzantine**

$m^S = \{m_{k,i,j}^S : k = 1, \dots, n, i = 0, \dots, t_b \text{ and } j = 0, \dots, t_b\}$

**Phase I: S to R:**

1. Using the  $m_{k,i,j}$  values, **S** defines  $n$  bivariate polynomials  $F_k^S(x, y), k = 1, \dots, n$  as follows:  

$$F_k^S(x, y) = \sum_{i=0, j=0}^{i=t_b, j=t_b} m_{k,i,j}^S x^i y^j.$$
2. **S** then evaluates each  $F_k^S(x, y), k = 1, \dots, n$  at  $y = 1, \dots, n$  to obtain total  $n^2$  univariate polynomials denoted as  $p_{k,i}^S(x), 1 \leq k, i \leq n$ , each of degree  $t_b$  where  $p_{k,i}^S(x) = F_k^S(x, i)$ .
3. For  $i = 1, \dots, n$ , **S** sends the following to **R** over wire  $w_i$ :
  - (a) Polynomials  $p_{k,i}^S(x), k = 1, \dots, n$ ;
  - (b) Values  $r_{k,j,i}^S$ , for  $1 \leq k, j \leq n$ , where  $r_{k,j,i}^S = p_{k,j}^S(i)$ .

**Phase II: R to S:**

1. Let **R** receive the following over wire  $w_i$ , for  $i = 1, \dots, n$  (see Fig. 4.1 for the pictorial representation):
  - (a) Polynomials  $p_{k,i}^R(x), k = 1, \dots, n$ ;
  - (b) Values  $r_{k,j,i}^R$ , for  $1 \leq k, j \leq n$ .
2. Using the received values, **R** constructs  $n$  conflict graphs  $H_1, \dots, H_n$ . **R** then combines these graphs into a single conflict graph  $H^R$  using **Union Technique**, as explained in the previous section.
3. **R** constructs the list of seven tuples  $X$  corresponding to  $H^R$  (as explained in the previous section) and broadcasts  $X$  to **S**.

**Phase III: S to R:**

1. **S** reliably receives the list  $X$ . **S** then creates a list  $L_{fault}$  which is initialized to  $\emptyset$ .
2. For each seven tuple  $\{w_i, w_j, \gamma, p_{\gamma,i}^R(j), r_{\gamma,i,j}^R, p_{\gamma,j}^R(i), r_{\gamma,j,i}^R\} \in X$ , **S** does the following:
  - (a) **S** checks  $p_{\gamma,i}^R(j) \stackrel{?}{=} p_{\gamma,i}^S(j)$ . If not, then **S** adds  $w_i$  to  $L_{fault}$ .
  - (b) **S** checks  $r_{\gamma,i,j}^R \stackrel{?}{=} r_{\gamma,i,j}^S$ . If not, then **S** adds  $w_j$  to  $L_{fault}$ .
  - (c) **S** checks  $p_{\gamma,j}^R(i) \stackrel{?}{=} p_{\gamma,j}^S(i)$ . If not, then **S** adds  $w_j$  to  $L_{fault}$ .
  - (d) **S** checks  $r_{\gamma,j,i}^R \stackrel{?}{=} r_{\gamma,j,i}^S$ . If not, then **S** adds  $w_i$  to  $L_{fault}$ .
3. **S** finally broadcasts the list  $L_{fault}$  to **R** and terminates 3-Optimal-PRMT-Mobile-Byzantine.

**Message Recovery by R:**

1. **R** reliably receives  $L_{fault}$  and identifies all  $w_i$  over which it had received at least one incorrect polynomial  $p_{k,i}^R(x) \neq p_{k,i}^S(x)$  during **Phase I**.
2. **R** neglects all the polynomials  $p_{k,i}^R(x), k = 1, \dots, n$ , received over each  $w_i \in L_{fault}$  during **Phase I**.
3. Using the remaining (at least  $t_b + 1$ )  $p_{k,i}^R(x)$ 's, **R** correctly recovers the bivariate polynomial  $F_k^S(x, y)$ , for  $k = 1, \dots, n$  and hence the message  $m^S$ .

PROOF: During **Phase I**, **S** sends over each wire  $n$  polynomials of degree  $t_b$  and  $n^2$  values. So communication complexity of **Phase I** is  $\mathcal{O}(n^3)$ . During **Phase II**, **R** broadcasts the list  $X$ . As explained earlier,  $X$  contains  $\mathcal{O}(n^2)$  tuples. Hence broadcasting  $X$  requires  $\mathcal{O}(n^3)$  communication complexity. During **Phase III**, **S** broadcasts the list  $L_{fault}$ . Since  $|L_{fault}| \leq t_b$ , this involves communicating  $\mathcal{O}(nt_b) = \mathcal{O}(n^2)$  field

elements. Hence the overall communication complexity of protocol 3-Optimal-PRMT-Mobile-Byzantine is  $\mathcal{O}(n^3)$ .  $\square$

**Theorem 4.7** *Protocol 3-Optimal-PRMT-Mobile-Byzantine is a communication optimal PRMT protocol which achieves reliability with constant factor overhead tolerating  $\mathcal{A}_{t_b}^{mobile}$ .*

PROOF: From Theorem 4.3, any three phase PRMT protocol over  $n = 2t_b + 1$  wires must communicate  $\Omega\left(\frac{n\ell}{n-t_b}\right) = \Omega(\ell)$  field elements to reliably send a message containing  $\ell$  field elements against  $\mathcal{A}_{t_b}^{mobile}$ . This is because  $n = 2t_b + 1$  and hence  $n - t_b = \Theta(n)$ . Now substituting  $\ell = n(t_b + 1)^2$ , we find that any three phase PRMT protocol over  $n = 2t_b + 1$  wires has to communicate  $\Omega(nt_b^2) = \Omega(n^3)$  field elements to reliably send a message containing  $\Theta(n^3)$  field elements. From Lemma 4.6, the total communication complexity of protocol 3-Optimal-PRMT-Mobile-Byzantine is  $\mathcal{O}(n^3)$ . Hence the protocol is communication optimal. Moreover, it is easy to see that the protocol achieves reliability with constant factor overhead.  $\square$

**Theorem 4.8** *Let  $\mathcal{N}$  be an undirected synchronous network, under the influence of  $\mathcal{A}_{t_b}^{mobile}$ , where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + 1$  wires. Then three phases are necessary and sufficient for the existence of any PRMT protocol which achieves reliability with constant factor overhead.*

PROOF: Follows from Theorem 4.3 and Theorem 4.7.  $\square$

## 4.5 PRMT Tolerating Mobile Adversary (in Terms of Rounds)

Till the previous section, we concentrated to design communication optimal PRMT protocol, when the network is abstracted in terms of wires. The merits of working in such a model are as follows:

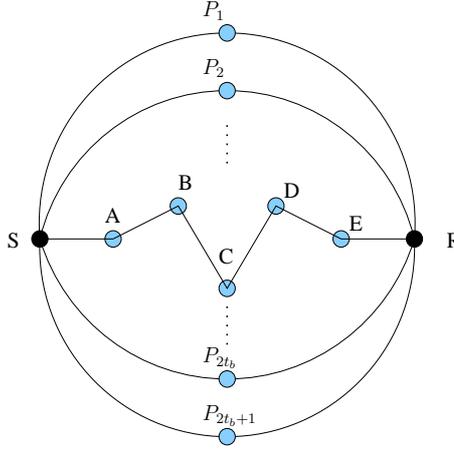
1. It eases deriving the connectivity requirement for the POSSIBILITY of PRMT/PSMT protocols and also the lower bounds for the communication complexity of protocols.
2. It simplifies the analysis of any protocol designed in such model.

But this model has its own demerits. In many practical scenarios, modelling the network as wires, does not give correct estimation of the communication complexity of PRMT protocols. To understand the statement, we provide a motivating example. Consider the network consisting of  $(2t_b + 8)$  vertices, as given in Fig. 4.3.

Suppose the network in Fig. 4.3 is abstracted as a collection of  $(2t_b + 2)$  wires, under the control of  $\mathcal{A}_{t_b}^{mobile}$ . It is easy to see that in this network, there exists a single phase PRMT protocol (namely broadcast protocol), which reliably sends a message containing  $\ell$  field elements by communicating  $\mathcal{O}(n\ell)$  field elements, tolerating  $\mathcal{A}_{t_b}^{mobile}$ .

Now suppose that the protocol execution take place in a sequence of *rounds*, where at the beginning of each round, each node send messages to their neighbors. Thus, the messages sent by a node in round  $k$ , reaches its neighbor at the beginning of round  $k + 1$ . Then the so called single phase broadcast protocol runs for *six* rounds (which is the length of the longest path), with a communication complexity of  $\mathcal{O}(n)$  times the message size. Now the question is whether there exists a six round PRMT protocol in the network of Fig. 4.3 with a better communication complexity. The answer is yes!

Figure 4.3: A  $(2t_b + 2)$ -(**S**,**R**)-connected Network.



Consider the following protocol: **S** and **R** run the three phase PRMT protocol 3-Optimal-PRMT-Mobile-Byzantine using the wires along the nodes  $P_1, P_2, \dots, P_{2t_b+1}$ , neglecting the path of length six (the longest path takes six rounds, while all other paths delivers message in two rounds). Thus while the single phase broadcast protocol has a complexity of  $\mathcal{O}(n\ell)$ , the three phase protocol has a communication complexity of  $\mathcal{O}(\ell)$ . Thus in Fig. 4.3, a six round protocol with a communication complexity of  $\mathcal{O}(\ell)$  is possible.

However, the information regarding the *length* of each of the paths (wires) in the actual network is completely lost in the wired abstraction. Thus in many practical scenarios, wired abstraction causes an over estimation in the round complexity and communication complexity of PRMT protocols in the original network. This motivates us to redefine our network model and adversary settings, which we do in the next section.

#### 4.5.1 Round Based Network and Adversary Settings

As shown in the previous section, it is necessary to use more fine-grained and hence stronger model, namely the graph based one (in comparison to the collection of wires) for designing and analyzing *optimal* PRMT protocols. So we consider a graph with internal details in the following way. Let  $H$  be an undirected graph under the control of  $\mathcal{A}_{t_b}^{mobile}$ . From Theorem 4.1,  $H$  should be  $(2t_b + 1)$ -(**S**, **R**) connected which is a necessary and sufficient condition for PRMT tolerating  $\mathcal{A}_{t_b}^{mobile}$ . Let  $G$  be the subgraph of  $H$  induced by the  $2t_b + 1$  vertex disjoint paths between **S** and **R**. If there are more than  $2t_b + 1$  vertex disjoint paths in  $H$ , then  $G$  will also contain these paths. In the following sections, we work on  $G$  to derive tight lower bound on round complexity for reliable communication and design protocols on  $G$ .

The system is assumed to be *synchronous*. Any protocol is assumed to be executed in a sequence of *rounds*, wherein in each round, a node perform some local computation, sends new messages to his out-neighbors and receive the messages sent in previous round by his in-neighbors. The distrust in the network is modelled by a mobile Byzantine adversary  $\mathcal{A}_{t_b}^{mobile}$ . The behavior of  $\mathcal{A}_{t_b}^{mobile}$  is re-defined to allow it to corrupt any set of  $t_b$  nodes, after every  $\rho \geq 1$  rounds, where  $\rho$  is called the *roaming speed* of the adversary. We first consider the worst case, that of  $\rho = 1$ . Later on, we extend our results to arbitrary value of  $\rho$ .

More formally, before the beginning of round  $k$ , the adversary can corrupt any set of nodes  $\mathcal{P}_{corrupt}$ , consisting of  $t_b$  nodes. Then the adversary has access to the messages sent to the nodes in  $\mathcal{P}_{corrupt}$  in round  $k - 1$  and can alter the behavior of the nodes in  $\mathcal{P}_{corrupt}$  arbitrarily in round  $k$ . However by corrupting a node  $P$  in round  $k$  the adversary does not obtain information about the messages to and from the node  $P$  in all the previous rounds, i.e., the protocol can choose to delete some information from the (honest) node at the end of a round, to make sure that the information is not available to the adversary even if he corrupts the node at a later round. Before computing the minimum number of rounds for reliable communication, we explain the concept of transmission graph.

#### 4.5.2 Transmission Graph

Graphs always have been used as a very powerful abstraction of the network by modelling the physical link between two nodes as an edge between the corresponding vertices of the graph. However it does not contain any temporal information. Especially in the case of a mobile adversary, where the adversary can corrupt different set of nodes at different time instance, a graph representation of the network is inadequate. However since the protocol itself discretizes time in terms of rounds, it is sufficient to model the system at each round, rather than each time instant. Hence, in [82], the authors have introduced the concept of transmission graph  $\mathcal{G}^d$  to study the execution of a protocol that has run for  $d$  rounds.

In the transmission graph  $\mathcal{G}^d$ , each node  $P$  is represented by a set of nodes  $\{P_0, P_1, P_2, \dots, P_d\}$ . The node  $P_r$  corresponds to the node  $P$  at round  $r$ . For any two neighboring nodes  $P$  and  $Q$  and any  $1 \leq r \leq d$ , a message sent by  $P$  to  $Q$  in round  $r - 1$  is available to  $Q$  only at round  $r$ . Hence there is an edge in  $\mathcal{G}^d$  connecting the node  $P_{r-1}$  to the node  $Q_r$  for all  $1 \leq r \leq d$ . Note that the transmission graph is a directed graph, because of the directed nature of time. So the edges between the nodes at consecutive time steps are always oriented towards increasing time. We now recall the definition of transmission graph from [82].

**Definition 4.9 (Transmission Graph [82])** *Given a graph  $G = (V, E)$  and a positive integer  $d$ , the transmission Graph  $\mathcal{G}^d$  is a directed graph defined as follows:*

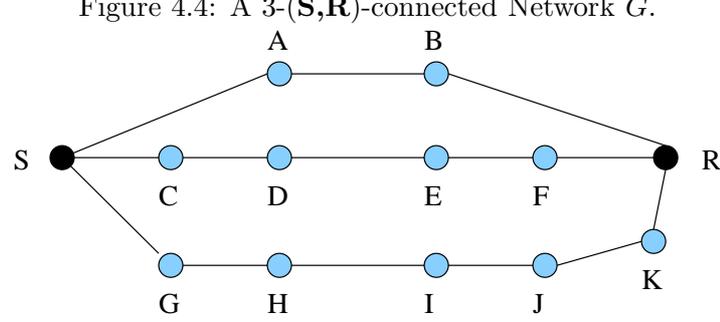
- Nodes of  $\mathcal{G}^d$  belong to  $V \times \{0 \dots d\}$  where the node  $(P, r) \in V \times \{0 \dots d\}$  is denoted by  $P_r$ .
- The edge set of  $\mathcal{G}^d$  is  $E^d = E_1 \cup E_2$  where  $E_1 = \{(P_{a_{r-1}}, P_{b_r}) \mid (P_a, P_b) \in E \text{ and } 1 \leq r \leq d\}$  and  $E_2 = \{(P_{a_{r-1}}, P_{a_r}) \mid P_a \in V \text{ and } 1 \leq r \leq d\}$ .

Let  $\mathcal{P}^r = \{P_{a_r} \mid P_a \in V\}$  and let  $\mathcal{ADV}_{mobile}$  be a threshold mobile adversary acting on the network  $G$ , that can corrupt any  $t_b$  nodes in each round. Consider an execution  $\Gamma$  of a  $d$ -round protocol on  $G$ . Suppose  $\mathcal{ADV}_{mobile}$  corrupts a set of nodes  $Adv_r = \{P_1, P_2, \dots, P_{t_b}\}$  in round  $r$  in  $G$ , then the same effect is obtained by corrupting the nodes  $Adv^r = \{P_{1_r}, P_{2_r}, \dots, P_{t_b_r}\}$  in  $\mathcal{G}^d$ . Hence the effect of  $\mathcal{ADV}_{mobile}$  on execution  $\Gamma$  can be simulated by a static general adversary, who corrupts  $\bigcup_{r=1}^d Adv^r$  on  $\mathcal{G}^d$ . More formally, we have the following lemma:

**Lemma 4.10** *Mobile adversary  $\mathcal{ADV}_{mobile}$  acting on the original network graph  $G$  for  $d$  rounds can be simulated by a static adversary given by the adversary structure  $\mathcal{ADV}_{static}^d = \{Adv^1 \cup Adv^2 \cup Adv^3 \dots \cup Adv^d \mid Adv^r \in \Pi_{t_b}(\mathcal{P}^r), 1 \leq r \leq d\}$  on  $\mathcal{G}^d$ , where*

$\Pi_{t_b}(\mathcal{P}^r)$  denotes the set of all subsets of cardinality  $t_b$  of the set  $\mathcal{P}^r$  excluding  $\mathbf{S}^r$  and  $\mathbf{R}^r$ .

**Example 4.11** Consider the network shown in Fig. 4.4:



The network is 3-( $\mathbf{S}, \mathbf{R}$ )-connected and hence from Theorem 4.1, at most one mobile corruption (i.e.,  $\mathcal{A}_1^{\text{mobile}}$ ) can be tolerated by any PRMT protocol.

Now consider  $\mathcal{G}^4$ , where the adversary structure  $\mathcal{ADV}_{\text{static}}^4 = \{Adv^1 \cup Adv^2 \cup Adv^3 \cup Adv^4\}$ . Here each  $Adv^r \in \Pi_1(\mathcal{P}^r)$ ,  $1 \leq r \leq 4$ , where  $\Pi_1(\mathcal{P}^r)$  denotes the set of all subsets of cardinality one of the set  $\mathcal{P}^r$ . For example,  $\{A_1, A_2, A_3, A_4\}$ ,  $\{A_1, D_2, G_3, H_4\}$ ,  $\{H_1, E_2, B_3, A_4\}$  are some of the elements of  $\mathcal{ADV}_{\text{static}}^4$  in  $\mathcal{G}^4$ . Here  $\{A_1, A_2, A_3, A_4\}$  denotes an adversarial strategy where in the original network, the adversary corrupts the same node  $A$  in all the four rounds. Similarly  $\{H_1, E_2, B_3, A_4\}$  denotes an adversarial strategy where in the original network, the adversary corrupts the nodes  $H, E, B$  and  $A$  during first, second, third and fourth round respectively. In fact there are  $11^4$  possible elements of  $\mathcal{ADV}_{\text{static}}^4$  in  $\mathcal{G}^4$ , since there are 11 nodes in  $G$  (excluding  $\mathbf{S}$  and  $\mathbf{R}$ ) and in each of the four rounds, adversary can choose any one of the 11 nodes to corrupt. Now out of these  $11^4$  possible elements, only one element corresponds to the actual corruption for four rounds, that would have been done by the adversary on the original graph.

In general let  $G$  be a graph with  $2t_b + 1$  (or more) vertex disjoint paths between  $\mathbf{S}$  and  $\mathbf{R}$  and  $N$  be the total number of nodes in these paths. Then in  $\mathcal{G}^d$ , there will be  $\binom{N}{t_b+1}^d$  possible elements in the adversary structure  $\mathcal{ADV}_{\text{static}}^d$ . In order to find the minimum number of rounds for reliable communication, we slightly modify the definition of transmission graph as follows:

**Definition 4.12 (Modified Transmission Graph)** Given a graph  $G$  and an integer  $d > 0$ , the modified Transmission Graph  $G^d$  is the graph  $\mathcal{G}^d$  along with two additional nodes  $\mathcal{S}$  and  $\mathcal{R}$ , where  $\mathcal{S}$  is connected to all  $\mathbf{S}_r, 0 \leq r \leq d$  and each  $\mathbf{R}_r, 0 \leq r \leq d$  is connected to  $\mathcal{R}$ . Further the edges between  $(\mathbf{S}_{r-1}, \mathbf{S}_r)$  and  $(\mathbf{R}_{r-1}, \mathbf{R}_r)$  for  $1 \leq r \leq d$  are removed.

**Definition 4.13 (Securely Disjoint Paths)** Two paths  $\Gamma_1$  and  $\Gamma_2$  between the nodes  $\mathcal{S}$  and  $\mathcal{R}$  in the modified transmission graph  $G^d$  are said to be securely disjoint if the common nodes (if any) between the two paths are only of the type  $\mathbf{S}_a$  and  $\mathbf{R}_b$  for some value of  $a$  and  $b$ . That is,  $\Gamma_1 \cap \Gamma_2 \subset \{\mathbf{S}_0, \mathbf{S}_1, \mathbf{S}_2 \dots \mathbf{S}_d\} \cup \{\mathbf{R}_0, \mathbf{R}_1, \mathbf{R}_2 \dots \mathbf{R}_d\}$ .

**Definition 4.14 (Space Time Path)** Given a path  $\Gamma = \{\mathbf{S}, P_1, P_2 \dots P_z, \mathbf{R}\}$  from  $\mathbf{S}$  to  $\mathbf{R}$  in the original graph  $G$ , the space-time path  $\Gamma^i$  in graph  $G^d$  is defined as  $\Gamma^i = \{\mathcal{S}, \mathbf{S}_i, P_{1+i}, P_{2+i}, \dots P_{z+i}, \mathbf{R}_{i+z+1}, \mathcal{R}\}$ ,  $0 \leq i \leq d - z - 1$ .

We now illustrate the above definitions with the following example:

**Example 4.15** Consider the path  $\Gamma = \{\mathbf{S}, A, B, \mathbf{R}\}$  in Fig. 4.4. Now in  $G^5$ , there are three space time paths corresponding to the path  $\Gamma$ , namely  $\Gamma^0 = \{\mathcal{S}, \mathbf{S}_0, A_1, B_2, \mathbf{R}_3, \mathcal{R}\}$ ,  $\Gamma^1 = \{\mathcal{S}, \mathbf{S}_1, A_2, B_3, \mathbf{R}_4, \mathcal{R}\}$  and  $\Gamma^2 = \{\mathcal{S}, \mathbf{S}_2, A_3, B_4, \mathbf{R}_5, \mathcal{R}\}$ . The space time path  $\Gamma^0$  can be interpreted as  $\mathbf{S}$  communicating to  $A$  in the 0<sup>th</sup> round,  $A$  communicating to  $B$  in the first round,  $B$  communicating to  $\mathbf{R}$  in the second round which is received by  $\mathbf{R}$  in the third round. Similarly, the paths  $\Gamma^1$  and  $\Gamma^2$  can be interpreted. Note that in  $G^5$ , there are only three space time paths corresponding to the path  $\Gamma$  in  $G$ . This is so because if any protocol is executed for five rounds, then  $\mathbf{R}$  will stop receiving anything from  $B$  after fifth round. In general, let  $G$  be a graph and  $\Gamma$  be a path between  $\mathbf{S}$  and  $\mathbf{R}$  containing  $z$  nodes (i.e., the path length is  $z + 1$ ). Then in the transmission graph  $G^d, d > z$ , there will be  $d - z$  space time paths corresponding to the path  $\Gamma$ , namely  $\Gamma^i, 0 \leq i \leq d - z - 1$ .

**Lemma 4.16 ([82])** For any path  $\Gamma$  of length  $z$  (containing  $z + 1$  nodes) from  $\mathbf{S}$  to  $\mathbf{R}$  in  $G$ , the paths  $\Gamma^i, 0 \leq i \leq d - z$  are pairwise securely disjoint. Further, for any two vertex disjoint paths  $\Gamma_1, \Gamma_2$  in  $G$  and for any  $i, j$ , the paths  $\Gamma_1^i$  and  $\Gamma_2^j$  are securely disjoint.

We illustrate the above lemma with the following example:

**Example 4.17** Consider the paths  $\Gamma_1 = \{\mathbf{S}, A, B, \mathbf{R}\}$  and  $\Gamma_2 = \{\mathbf{S}, C, D, E, F, \mathbf{R}\}$  in the network shown in Fig. 4.4. Suppose we consider the transmission graph  $G^6$ , then there are following space time paths corresponding to  $\Gamma_1$  in  $G^6$ :

1.  $\Gamma_1^0 = \{\mathcal{S}, \mathbf{S}_0, A_1, B_2, \mathbf{R}_3, \mathcal{R}\}$ ,
2.  $\Gamma_1^1 = \{\mathcal{S}, \mathbf{S}_1, A_2, B_3, \mathbf{R}_4, \mathcal{R}\}$ ,
3.  $\Gamma_1^2 = \{\mathcal{S}, \mathbf{S}_2, A_3, B_4, \mathbf{R}_5, \mathcal{R}\}$  and
4.  $\Gamma_1^3 = \{\mathcal{S}, \mathbf{S}_3, A_4, B_5, \mathbf{R}_6, \mathcal{R}\}$ .

Similarly, there are following space time paths corresponding to  $\Gamma_2$  in  $G^6$ :

1.  $\Gamma_2^0 = \{\mathcal{S}, \mathbf{S}_0, C_1, D_2, E_3, F_4, \mathbf{R}_5, \mathcal{R}\}$  and
2.  $\Gamma_2^1 = \{\mathcal{S}, \mathbf{S}_1, C_2, D_3, E_4, F_5, \mathbf{R}_6, \mathcal{R}\}$ .

It is clear that all  $\Gamma_1^i, 0 \leq i \leq 3$  are securely disjoint. Similarly, all  $\Gamma_2^i, 0 \leq i \leq 1$  are securely disjoint. Also all the space time paths  $\Gamma_1^i, \Gamma_2^j, 0 \leq i \leq 3, 0 \leq j \leq 1$  are securely disjoint.

### 4.5.3 Computing Minimum Number of Rounds for PRMT with $\rho = 1$

In [82], the authors have computed a lower bound for the minimum number of rounds  $d$  for reliable communication from  $\mathbf{S}$  to  $\mathbf{R}$ . They showed that  $d > (2t_b + 1)N$  (see Lemma 4.1 of [82]), where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $2t_b + 1$  paths and  $N$  is the total number of nodes in the given network. However, we show that the bound in [82] is not tight. So, we derive tight bound on the minimum number of rounds, denoted by  $r_{min}$ , required for reliable communication from  $\mathbf{S}$  to  $\mathbf{R}$ .

Consider a graph  $G$  where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $2t_b + 1$  vertex disjoint paths  $\{\Gamma_1, \Gamma_2, \dots, \Gamma_{2t_b+1}\}$ . Without loss of generality, assume that the paths are arranged in

ascending order of path length. Let  $N_i$  denote the number of nodes in  $\Gamma_i$ ,  $1 \leq i \leq 2t_b + 1$ . Then as explained earlier, in  $G^d$ , there will be  $d - N_i$  space time paths, corresponding to the path  $\Gamma_i$  in  $G$ , provided  $d - N_i > 0$ . If  $d - N_i \leq 0$  then there will be no space time path corresponding to  $\Gamma_i$  in  $G^d$ . Assuming that each of the term  $d - N_i$  is positive, the total number of the space time paths in  $G^d$  is  $\sum_{i=1}^{i=2t_b+1} (d - N_i)$ . From Lemma 4.16, all these paths are securely disjoint. Now if any reliable protocol is executed on the original graph  $G$  for  $d$  rounds, then the adversary can make corruption only up to  $(d-1)$  rounds because in any reliable protocol, which is executed for  $d$  rounds,  $\mathbf{R}$  will receive information from its neighboring nodes in round  $d$ , which they sent to  $\mathbf{R}$  in round  $d-1$  and terminates the protocol. So even if adversary corrupts some node in round  $d$ , it will not effect the protocol, because the protocol will terminate in the  $d^{\text{th}}$  round itself. Note that if at least one node in a space time path in  $G^d$  is corrupted, it implies that the entire space time path is corrupted because the corrupted data introduced by the corrupted node will be forwarded by other nodes of the path in subsequent rounds. In general, since the adversary can corrupt at most  $t_b$  nodes in each round of any reliable protocol, it can corrupt at most  $t_b(d-1)$  nodes in  $G^d$  which can be distributed on  $t_b(d-1)$  secure disjoint paths in the worst case and hence each element in  $\mathcal{ADV}_{static}^d$  is of maximum cardinality  $t_b(d-1)$ . We now state the following theorem.

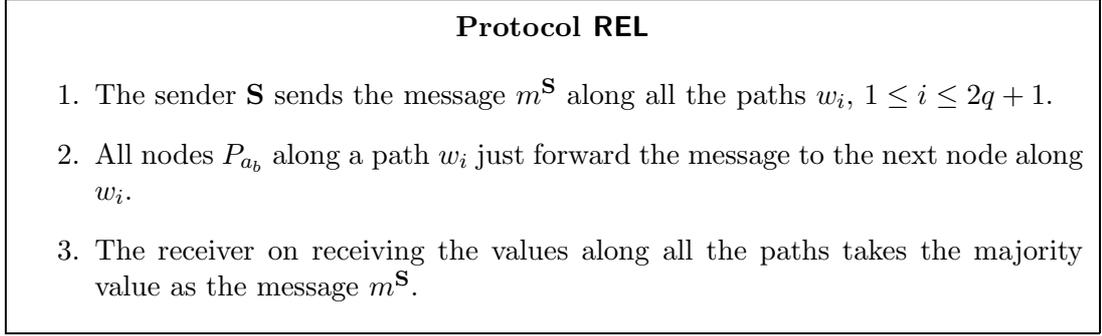
**Theorem 4.18** *Let  $G$  be an undirected network where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $2t_b + 1$  vertex disjoint paths  $\Gamma_1, \Gamma_2, \dots, \Gamma_{2t_b+1}$  with  $N_i$  nodes in  $\Gamma_i$ ,  $1 \leq i \leq 2t_b + 1$ . Let  $\mathcal{ADV}_{mobile}$  be a mobile adversary corrupting any set (probably different) of  $t_b$  nodes in each round of a protocol. Then the minimum number of rounds required for reliable communication from  $\mathbf{S}$  to  $\mathbf{R}$  is given by  $r_{min} = N - 2t_b + 1$ , where  $N = \sum_{i=1}^{i=2t_b+1} N_i$ .*

**PROOF: Necessity:** Let  $r_{min}$  be the minimum number of rounds required for reliable communication in  $G$ . Then as explained above, any mobile adversary  $\mathcal{ADV}_{mobile}$  can be simulated by a static adversary structure  $\mathcal{ADV}_{static}^{r_{min}}$  where each element of the adversary structure is of cardinality  $t_b(r_{min} - 1)$ . Also in  $G^{r_{min}}$ , there will be  $\sum_{i=1}^{i=2t_b+1} (r_{min} - N_i)$  securely disjoint paths between  $\mathcal{S}$  and  $\mathcal{R}$ , out of which at most  $t_b(r_{min} - 1)$  can be under the control of the adversary. Now it is known from [41], that reliable communication between  $\mathcal{S}$  and  $\mathcal{R}$  in a network in the presence of a static adversary given by an adversary structure is possible iff removal of the nodes belonging to any two adversarial sets in the adversary structure does not disconnect  $\mathcal{S}$  and  $\mathcal{R}$ . It implies that reliable communication in  $G$  under the presence of  $\mathcal{ADV}_{mobile}$  is possible in  $r_{min}$  rounds if  $\sum_{i=1}^{i=2t_b+1} (r_{min} - N_i) \geq 2t_b(r_{min} - 1) + 1$ . Solving this we get  $r_{min} \geq N - 2t_b + 1$  where  $N = \sum_{i=1}^{i=2t_b+1} N_i$ .

**Sufficiency:** Suppose  $r_{min} = N - 2t_b + 1$  where  $N = \sum_{i=1}^{i=2t_b+1} N_i$ . Then in  $G^{r_{min}}$  there are  $2t_b(r_{min} - 1) + 1$  securely disjoint paths from  $\mathbf{S}$  to  $\mathbf{R}$ , out of which at most  $t_b(r_{min} - 1)$  can be under the control of the adversary  $\mathcal{ADV}_{static}^{r_{min}}$ . Let us denote these paths by  $w_1, w_2, \dots, w_{2q+1}$ , where  $q = t_b(r_{min} - 1)$ . We now describe a reliable protocol called REL on the graph  $G^{r_{min}}$ , which takes  $r_{min}$  rounds. We also show how protocol REL can be executed on the real network  $G$  to reliably send  $m^{\mathbf{S}}$ . Protocol REL is given in Fig. 4.5.

Protocol REL can be emulated on  $G$  in the following way: if a node  $P_{1_b}$  and  $P_{2_{b+1}}$  are consecutive nodes in  $G^{r_{min}}$  along some path  $w_i$ , where  $w_i$  is the space time path corresponding to some physical path  $\Gamma_j$ ,  $1 \leq j \leq 2t_b + 1$ , then the node  $P_1$  on receiving  $\hat{m}^{\mathbf{S}}$  (possibly changed  $m^{\mathbf{S}}$ ) along the path  $\Gamma_j$  at the beginning of round  $b$  forwards it to the node  $P_2$  at the end of round  $b$  which is received by  $P_2$  in round  $b+1$ . The protocol has a communication complexity  $\mathcal{O}(2t_b(r_{min} - 1)|m^{\mathbf{S}}|)$  and this is polynomial in  $N$ .

Figure 4.5: An  $r_{min}$  Round PRMT Protocol on Graph  $G^{r_{min}}$



The correctness of the protocol is obvious. This completes the sufficiency proof.  $\square$

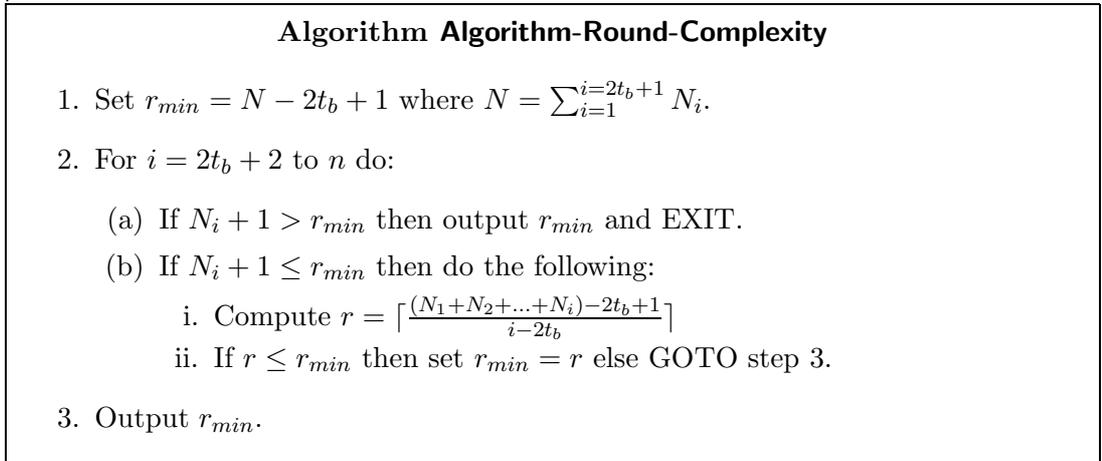
We now illustrate Theorem 4.18 with the following example:

**Example 4.19** For the network in Fig. 4.4,  $r_{min} = 10$ . This is because in  $G^9$ , there are total sixteen space time paths, of which the adversary can corrupt at most eight paths. So exactly half of the paths can be under the control of the adversary. However, in  $G^{10}$ , there are nineteen space time paths, out of which the adversary can corrupt at most nine paths. Hence, majority of the paths will be error free.

#### 4.5.4 Finding $r_{min}$ in the Presence of More than $2t_b + 1$ Paths for $\rho = 1$

In many practical scenarios there may be more than  $2t_b + 1$  vertex disjoint paths between **S** and **R**. Even then we can find  $r_{min}$  by using the same argument as in the previous section. Suppose  $G$  is a network where there are  $n$  node disjoint paths between **S** and **R**, where  $n > 2t_b + 1$ . Let the paths be denoted by  $\Gamma_1, \Gamma_2, \dots, \Gamma_{2t_b+1}, \dots, \Gamma_n$  and let the paths be arranged in ascending order of path length, such that there are  $N_i$  nodes in path  $\Gamma_i$ . We call the algorithm for computing  $r_{min}$  in this case as **Algorithm-Round-Complexity**, which is given in Fig. 4.6.

Figure 4.6: Algorithm for Finding  $r_{min}$  in the Presence of More than  $2t_b + 1$  Paths for  $\rho = 1$



The correctness of algorithm **Algorithm-Round-Complexity** is stated in the following theorem.

**Theorem 4.20** *Algorithm Algorithm-Round-Complexity correctly computes  $r_{min}$  when  $\mathbf{S}$  and  $\mathbf{R}$  are connected by more than  $2t_b + 1$  vertex disjoint paths and  $\rho = 1$ .*

PROOF: In Algorithm-Round-Complexity,  $r_{min}$  is first set to  $N - 2t_b + 1$ , which according to Theorem 4.18 is the minimum number of rounds for reliable communication in the presence of  $2t_b + 1$  paths  $\Gamma_j, 1 \leq j \leq 2t_b + 1$  between  $\mathbf{S}$  and  $\mathbf{R}$ . Note that the paths  $\Gamma_j, 1 \leq j \leq n$  are arranged in ascending order of path length. Now there are following cases to be considered for  $\Gamma_{2t_b+2}$ :

1.  $r_{min} < N_{2t_b+2} + 1$ : Note that  $N_{2t_b+2} + 1$  is the path length of  $\Gamma_{2t_b+2}$ . So if in any reliable protocol, the path  $\Gamma_{2t_b+2}$  is involved then it will take at least  $N_{2t_b+2} + 1$  rounds to send any information from  $\mathbf{S}$  to  $\mathbf{R}$  through the path  $\Gamma_{2t_b+2}$ . However, since  $r_{min} < N_{2t_b+2} + 1$ , including the path  $\Gamma_{2t_b+2}$  will increase  $r_{min}$ . Since the path lengths of remaining  $\Gamma_j, 2t_b + 3 \leq j \leq n$  is at least  $N_{2t_b+2} + 1$ , using the above argument, any round optimal protocol should only consider the first  $2t_b + 1$  paths and hence  $r_{min} = N - 2t_b + 1$ .
2.  $r_{min} \geq N_{2t_b+2} + 1$ : In this case, including  $\Gamma_{2t_b+2}$  may reduce the value of  $r_{min}$ . Using the argument of Theorem 4.18, we first compute minimum number of rounds  $r$  required for reliable communication considering the first  $2t_b + 2$  paths. Now  $r$  is computed by solving the inequality  $\sum_{i=1}^{i=2t_b+2} (r - N_i) \geq 2t_b(r - 1) + 1$  which implies  $r \geq \lceil \frac{(N_1 + N_2 + \dots + N_{2t_b+2}) - 2t_b + 1}{2} \rceil$ . If the minimum value of  $r$  is less than or equal to  $r_{min}$ , then considering  $\Gamma_{2t_b+2}$  reduces or does not change  $r_{min}$  and hence  $r_{min}$  is updated to  $r$ . Otherwise  $\Gamma_{2t_b+2}$  is neglected and  $r_{min}$  is not updated. However, if  $r > r_{min}$ , then including  $\Gamma_{2t_b+2}$  in any reliable protocol will increase  $r_{min}$ . Hence  $\Gamma_{2t_b+2}$  is not considered. Since the path lengths of remaining  $\Gamma_j, 2t_b + 3 \leq j \leq n$  is at least  $N_{2t_b+2} + 1$ , including any of them will increase  $r_{min}$ . Hence all of them are neglected.

In the algorithm, the above two checking is done for all  $\Gamma_i, 2t_b + 2 \leq i \leq n$ . Once  $r_{min}$  is computed,  $\mathbf{S}$  will know which paths to consider for reliably sending any message to  $\mathbf{R}$ . In the corresponding transmission graph  $G^{r_{min}}$  there will be  $2t_b(r_{min} - 1) + 1$  securely disjoint paths. So the protocol REL can be executed on  $G^{r_{min}}$  which can be simulated on original network  $G$  as specified in Theorem 4.18.  $\square$

We now illustrate Algorithm-Round-Complexity with the following example:

**Example 4.21** *Intuitively,  $r_{min}$  can be computed considering the first  $2t_b + 1$  shortest node paths between  $\mathbf{S}$  and  $\mathbf{R}$ . However, this is not always true! For example in Fig. 4.4, assuming  $t_b = 1$ , we find that  $r_{min} = 10$  (according to Theorem 4.18). Now if we add one more vertex disjoint path of six nodes between  $\mathbf{S}$  and  $\mathbf{R}$ , then from Algorithm-Round-Complexity,  $r_{min} = 8$ .*

#### 4.5.5 Computing $r_{min}$ for Arbitrary Roaming Speed

Till now we have considered a roaming speed of  $\rho = 1$ . We now consider a mobile adversary with roaming speed  $\rho > 1$  and compute  $r_{min}^\rho$ , which is the minimum number of rounds required for reliable communication from  $\mathbf{S}$  to  $\mathbf{R}$ , against a  $t_b$ -active mobile adversary, corrupting  $t_b$  nodes after every  $\rho$  rounds. Note that a mobile adversary with roaming speed one is the strongest adversary. Intuitively, reducing the roaming speed of the adversary will reduce the minimum number of rounds required for PRMT

between  $\mathbf{S}$  and  $\mathbf{R}$ . We support our intuition by computing  $r_{min}^\rho$  for an arbitrary  $\rho$ , where  $\rho > 1$ .

Let  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + 1$  vertex disjoint paths  $\Gamma_i, 1 \leq i \leq 2t_b + 1$ , which are in ascending order of path length and let  $\Gamma_i$  has  $N_i$  nodes. Without loss of generality, we assume that the adversary starts corruption from the first round of any protocol. Thus, if  $\rho = 2$  and if a protocol is executed for six rounds, then the adversary will corrupt  $t_b$  nodes in round one, three and five. Note that the  $t_b$  nodes which are corrupted in round one, three and five will also remain corrupted in the second, fourth and sixth round respectively. In general, any mobile adversary  $\mathcal{ADV}_{mobile}^\rho$  who corrupts any  $t_b$  nodes in the network after every  $\rho$  rounds in any  $r$  round protocol, can be simulated by a static adversary structure  $\mathcal{ADV}_{static}^\rho$  with size  $\binom{N}{t_b}^{\lceil \frac{r}{\rho} \rceil}$  where  $N$  is total number of nodes in  $2t_b + 1$  paths. This is because in  $r$  rounds, adversary will change the set of corrupted nodes after every  $\lceil \frac{r}{\rho} \rceil$  rounds.

We now show how the adversary with arbitrary roaming speed changes its control over space time paths with the help of following example. Recall that in  $G^r$ , each  $\Gamma_i, 1 \leq i \leq 2t_b + 1$  will have  $r - N_i$  securely disjoint space time paths.

**Example 4.22** Consider two space time paths  $\Gamma_1^0 = \{\mathcal{S}, \mathbf{S}_0, A_1, B_2, C_3, \mathbf{R}_4, \mathcal{R}\}$  and  $\Gamma_1^1 = \{\mathcal{S}, \mathbf{S}_1, A_2, B_3, C_4, \mathbf{R}_5, \mathcal{R}\}$  in  $G^5$ , corresponding to some path  $\Gamma_1 = \{\mathbf{S}, A, B, C, \mathbf{R}\}$  in a network  $G$ . If  $\rho = 1$  and if the adversary corrupts node  $A$  during first round, then  $\Gamma_1^0$  is corrupted. However, it does not imply that  $\Gamma_1^1$  is also corrupted until and unless the adversary corrupts node  $A$  in the second round also. However, if  $\rho = 2$  and if the adversary corrupts node  $A$  during the first round, then both  $\Gamma_1^0$  and  $\Gamma_1^1$  will be corrupted because node  $A$  will remain corrupted during the second round also. Thus for  $\rho = 1$ , the two space time paths are independent of each other but for  $\rho = 2$ , the two space time paths can be treated as one set, which will be corrupted if the adversary corrupts the first node of the physical path during the first round.

In general, if any reliable protocol is executed for  $r$  rounds, then in  $G^r$ , each  $\Gamma_i, 1 \leq i \leq 2t_b + 1$  will have  $\lceil \frac{r - N_i}{\rho} \rceil$  independent securely disjoint set of space time paths. Notice that if  $\rho = 1$ , then each space time path is itself an independent set and hence we get  $r - N_i$  independent sets for each  $\Gamma_i$ . Since the adversary can corrupt nodes up to  $r - 1$  rounds, in  $G^r$ , at most  $\lceil \frac{r-1}{\rho} \rceil \times t_b$  independent sets can be corrupted. This is because out of  $r - 1$  rounds, the adversary will change the set of corrupted nodes  $\lceil \frac{r-1}{\rho} \rceil$  times. We now state the following theorem:

**Theorem 4.23** Let  $G$  be a  $(2t_b + 1)$ - $(\mathbf{S}, \mathbf{R})$  connected undirected network under the influence of a  $t_b$ -active mobile adversary with roaming speed of  $\rho > 1$ . Then the minimum number of rounds  $r_{min}^\rho$  required for reliable communication from  $\mathbf{S}$  to  $\mathbf{R}$  in  $G$  is given by  $r_{min}^\rho = \min \{r, r_{min}^{\rho-1}\}$  where  $r$  is the minimum value satisfying the inequality  $\sum_{i=1}^{i=2t_b+1} \lceil \frac{r - N_i}{\rho} \rceil \geq 2 \lceil \frac{r-1}{\rho} \rceil \times t_b + 1$ .

**PROOF: Necessity:** In  $G^r$ , there will be  $\sum_{i=1}^{i=2t_b+1} \lceil \frac{r - N_i}{\rho} \rceil$  independent set of securely disjoint paths, out of which at most  $\lceil \frac{r-1}{\rho} \rceil t_b$  independent sets could be corrupted. Considering each independent set as wires, from [28],  $r$  will be  $r_{min}^\rho$  if  $\sum_{i=1}^{i=2t_b+1} \lceil \frac{r - N_i}{\rho} \rceil \geq 2 \lceil \frac{r-1}{\rho} \rceil \times t_b + 1$ . If the minimum value of  $r$  satisfying this inequality is greater than  $r_{min}^{\rho-1}$ , then  $r_{min}^\rho = r_{min}^{\rho-1}$  because a mobile adversary with roaming speed  $\rho$  is always weaker in capability than one with roaming speed  $\rho - 1$ . Hence any round optimal PRMT protocol tolerating a mobile adversary with roaming speed  $\rho - 1$  can always

withstand a mobile adversary with roaming speed  $\rho$ .

**Sufficiency:** We design PRMT protocol called  $\text{REL}^\rho$ , which reliably sends a message from  $\mathbf{S}$  to  $\mathbf{R}$  in  $r_{\min}^\rho$  rounds. If  $r_{\min}^\rho = r_{\min}^{\rho-1}$ , then  $\text{REL}^\rho$  is same as  $\text{REL}^{\rho-1}$ . Otherwise,  $\text{REL}^\rho$  is defined as given in Fig. 4.7.

Figure 4.7: Round-Optimal Reliable Message transmission of  $m^{\mathbf{S}}$  Tolerating a Mobile Adversary with Roaming Speed of  $\rho > 1$ .

<b>Protocol <math>\text{REL}^\rho</math></b>
<ol style="list-style-type: none"> <li>1. <math>\mathbf{S}</math> sends <math>m^{\mathbf{S}}</math> along the first space time path of each <math>\sum_{i=1}^{i=2t_b+1} \lceil \frac{r_{\min}^\rho - N_i}{\rho} \rceil</math> independent set of securely disjoint space time paths.</li> <li>2. <math>\mathbf{R}</math> only considers the values received along the first space time path of each of the <math>\sum_{i=1}^{i=2t_b+1} \lceil \frac{r_{\min}^\rho - N_i}{\rho} \rceil</math> independent set of securely disjoint space time paths and outputs the majority as <math>m^{\mathbf{S}}</math>.</li> </ol>

The correctness of protocol  $\text{REL}^\rho$  follows from the fact that  $\mathbf{R}$  will receive  $\sum_{i=1}^{i=2t_b+1} \lceil \frac{r_{\min}^\rho - N_i}{\rho} \rceil$  different copies of the message  $m^{\mathbf{S}}$ , out of which at most  $\lceil \frac{r_{\min}^\rho - 1}{\rho} \rceil \times t_b$  can be corrupted. However since  $\sum_{i=1}^{i=2t_b+1} \lceil \frac{r_{\min}^\rho - N_i}{\rho} \rceil \geq 2 \lceil \frac{r_{\min}^\rho - 1}{\rho} \rceil \times t_b + 1$ ,  $\mathbf{R}$  will always receive the correct message  $m^{\mathbf{S}}$  along the majority of the paths.  $\square$

We now illustrate Theorem 4.23 with the following example:

**Example 4.24** Consider the network  $G$  in Fig. 4.4. If  $\rho = 1$ , then from Theorem 4.18,  $r_{\min}^1 = 10$ . However, if  $\rho = 2$ , then from Theorem 4.23,  $r_{\min}^2 = 9$ . For the network  $G$  of Fig. 4.4, in the transmission graph  $G^8$ , there will be the following space time paths between  $\mathbf{S}$  and  $\mathbf{R}$ :

1.  $\Gamma_1^0 = \{\mathcal{S}, \mathbf{S}_0, A_1, B_2, \mathbf{R}_3, \mathcal{R}\};$
2.  $\Gamma_1^1 = \{\mathcal{S}, \mathbf{S}_1, A_2, B_3, \mathbf{R}_4, \mathcal{R}\};$
3.  $\Gamma_1^2 = \{\mathcal{S}, \mathbf{S}_2, A_3, B_4, \mathbf{R}_5, \mathcal{R}\};$
4.  $\Gamma_1^3 = \{\mathcal{S}, \mathbf{S}_3, A_4, B_5, \mathbf{R}_6, \mathcal{R}\};$
5.  $\Gamma_1^4 = \{\mathcal{S}, \mathbf{S}_4, A_5, B_6, \mathbf{R}_7, \mathcal{R}\};$
6.  $\Gamma_1^5 = \{\mathcal{S}, \mathbf{S}_5, A_6, B_7, \mathbf{R}_8, \mathcal{R}\};$
7.  $\Gamma_2^0 = \{\mathcal{S}, \mathbf{S}_0, C_1, D_2, E_3, F_4, \mathbf{R}_5, \mathcal{R}\};$
8.  $\Gamma_2^1 = \{\mathcal{S}, \mathbf{S}_1, C_2, D_3, E_4, F_5, \mathbf{R}_6, \mathcal{R}\};$
9.  $\Gamma_2^2 = \{\mathcal{S}, \mathbf{S}_2, C_3, D_4, E_5, F_6, \mathbf{R}_7, \mathcal{R}\};$
10.  $\Gamma_2^3 = \{\mathcal{S}, \mathbf{S}_3, C_4, D_5, E_6, F_7, \mathbf{R}_8, \mathcal{R}\};$
11.  $\Gamma_3^0 = \{\mathcal{S}, \mathbf{S}_0, G_1, H_2, I_3, J_4, K_5, \mathbf{R}_6, \mathcal{R}\};$
12.  $\Gamma_3^1 = \{\mathcal{S}, \mathbf{S}_1, G_2, H_3, I_4, J_5, K_6, \mathbf{R}_7, \mathcal{R}\};$

$$13. \Gamma_3^2 = \{\mathcal{S}, \mathbf{S}_2, G_3, H_4, I_5, J_6, K_7, \mathbf{R}_8, \mathcal{R}\}.$$

If  $\rho = 2$ , then there will be total seven independent set of securely disjoint paths (three corresponding to  $\Gamma_1$ , two corresponding to  $\Gamma_2$  and two corresponding to  $\Gamma_3$ ). Note that the last set of securely disjoint path corresponding to  $\Gamma_3$  will have only one path unlike the other sets, each of which will have two paths. Now out of the eight rounds, adversary can do corruption in round one, three, five and seven. Hence, there can be at most four sets of securely disjoint paths out of the seven sets which can be under the control of the adversary. Since majority of the sets will be under the control of the adversary, no reliable protocol is possible in eight rounds. More formally, there exists two elements in the the static adversary structure  $\mathcal{ADV}_{static}^8$  corresponding to the transmission graph  $G^8$ , such that removal of all the securely disjoint paths passing through these nodes in  $G^8$  disconnects  $\mathcal{S}$  and  $\mathcal{R}$ . For example, consider the sets  $\{A_1, A_2, A_3, A_4, A_5, A_6, K_7, K_8\}$  and  $\{C_1, C_2, C_3, C_4, K_5, K_6, K_7, K_8\}$  belonging to the adversary structure  $\mathcal{ADV}_{static}^8$ . The set  $\{A_1, A_2, A_3, A_4, A_5, A_6, K_7, K_8\}$  denotes an adversary who corrupts nodes  $A$  in the first round (and hence in the second round also because  $\rho = 2$ ), node  $A$  in the third round (and hence in the fourth round also), node  $A$  in the fifth round (and hence in the sixth round also) and finally node  $K$  in the seventh round. Similarly, the other adversary element can be interpreted. Now it is easy to see that all the space time paths in  $G^8$  passes through one of the nodes in  $\{A_1, A_2, A_3, A_4, A_5, A_6, K_7, K_8\} \cup \{C_1, C_2, C_3, C_4, K_5, K_6, K_7, K_8\}$ . Hence removal of these nodes will disconnect  $\mathcal{S}$  and  $\mathcal{R}$  and hence no reliable protocol will exist in  $G^8$  and hence  $r_{min}^2 \neq 8$ .

However, if we consider the transmission graph  $G^9$ , then there will be nine independent set of securely disjoint paths between  $\mathbf{S}$  and  $\mathbf{R}$  (four corresponding to  $\Gamma_1$ , three corresponding to  $\Gamma_2$  and two corresponding to  $\Gamma_3$ ), out of which at most four sets can be under the control of the adversary. Hence majority of the sets will not be under the control of the adversary and hence reliable protocol is possible between  $\mathbf{S}$  and  $\mathbf{R}$  in  $G^9$ . Since the protocol can be simulated in the original network  $G$  in nine rounds,  $r_{min}^2 = 9$ . Note that for  $G$ ,  $r_{min}^1 = 10$ . Hence  $r_{min}^2 < r_{min}^1$ .

Once we know how to compute  $r_{min}^\rho$  in the presence of  $2t_b + 1$  node disjoint paths between  $\mathbf{S}$  and  $\mathbf{R}$ , Algorithm-Round-Complexity can be adapted to find  $r_{min}^\rho$  in the presence of more than  $2t_b + 1$  node disjoint paths between  $\mathbf{S}$  and  $\mathbf{R}$ , tolerating a  $t_b$ -active mobile adversary with roaming speed of  $\rho > 1$ .

#### 4.5.6 Computing Minimum Number of Rounds for Static Adversary

Here we compute  $r_{min}$  for reliable communication against a  $t_b$ -active static adversary. If a node is corrupted by the static adversary in some round, then it remains corrupted for the remaining rounds of the protocol. Hence, the total number of nodes that will be corrupted throughout the protocol is  $t_b$ .

**Theorem 4.25** *Let  $G$  be a  $(2t_b + 1)$ - $(\mathbf{S}, \mathbf{R})$  connected undirected network under the influence of a  $t_b$ -active static adversary. Let  $\Gamma_1, \Gamma_2, \dots, \Gamma_{2t_b+1}$  be the  $2t_b + 1$  vertex disjoint paths with  $N_i$  nodes in  $\Gamma_i, 1 \leq i \leq 2t_b + 1$ . Let the paths be arranged in ascending order of path length. Then  $r_{min} = N_{2t_b+1} + 1$ , the length of the longest path  $\Gamma_{2t_b+1}$ .*

**PROOF: Necessity:** A node once corrupted by static adversary remains so for the remaining rounds of the protocol. Hence all the space time paths passing through the node remain corrupted. Thus, if the adversary corrupts the first node of  $\Gamma_i$  during the

first round of an  $r$  round PRMT protocol, then all the  $r - N_i + 1$  space time paths  $\Gamma_i^j, 0 \leq j \leq r - N_i$  will be corrupted (this is the worst adversary strategy). So all these paths can be considered as a single set controlled by the adversary. Likewise, all the individual space time paths corresponding to each  $\Gamma_i$  can be considered as a single set. Hence  $r_{min}$  is the minimum value of  $r$  such that after  $r$  rounds, there exists  $2t_b + 1$  such independent sets (corresponding to each of the  $2t_b + 1$  physical paths in  $G$ ). It is easy to verify that  $r_{min}$  is  $N_{2t_b+1} + 1$  which is the length of the longest path  $\Gamma_{2t_b+1}$  in  $G$ . The reason is that the independent set corresponding to  $\Gamma_{2t_b+1}$  will be generated only in  $G^{N_{2t_b+1}+1}$ ; i.e., after  $N_{2t_b+1} + 1$  rounds. Before that, in  $G^r, r = N_{2t_b+1}$ , only the independent sets corresponding to  $\Gamma_i, 1 \leq i \leq 2t_b$  will be generated. Hence there will be only  $2t_b$  such independent sets in  $G^{N_{2t_b+1}}$ , out of which at most  $t_b$  can be corrupted by the adversary. Hence no reliable protocol will be possible in  $G^{N_{2t_b+1}}$  and hence  $r_{min}$  will be at least  $N_{2t_b+1} + 1$ .

**Sufficiency:** Consider the following protocol in  $G^{N_{2t_b+1}+1}$ : **S** sends the message along the first space time path corresponding to each of the  $2t_b + 1$  independent sets. On receiving, **R** will output majority as the message. The correctness of the protocol follows from the fact that in  $G^{N_{2t_b+1}+1}$ , there will be  $2t_b + 1$  independent sets of paths, of which at most  $t_b$  could be corrupted.  $\square$

#### 4.5.7 Communication Optimal PRMT Protocol in Terms of Rounds

Let **S** and **R** be connected by  $2t_b + 1$  node disjoint paths and let there be a  $t_b$ -active mobile adversary, with roaming speed of  $\rho = 1$ . From Theorem 4.18, in  $G^{r_{min}}$  there will be  $2t_b(r_{min} - 1) + 1$  securely disjoint paths, out of which at most  $t_b(r_{min} - 1)$  can be corrupted. However each of these paths are temporal and hence can be used at most once. We now present the modified version of three phase PRMT protocol 3-Optimal-PRMT-Mobile-Byzantine, called PRMT-Round-Mobile-Byzantine, tolerating a mobile adversary who can corrupt  $t_b$  nodes in every round of a protocol.

PRMT-Round-Mobile-Byzantine is executed for  $3r_{min}$  rounds on  $G$ , where  $G$  is the original network consisting  $2t_b + 1$  vertex disjoint paths between **S** and **R**. The first phase of 3-Optimal-PRMT-Mobile-Byzantine is executed in the first  $r_{min}$  rounds from **S** to **R**, the second phase of 3-Optimal-PRMT-Mobile-Byzantine is executed in the next  $r_{min}$  rounds from **R** to **S** and finally the third phase in the last  $r_{min}$  rounds from **S** to **R**. This can be visualized as executing a  $3r_{min}$  round protocol on  $G^{3r_{min}}$ , where first  $r_{min}$  rounds are executed from **S** to **R**, next  $r_{min}$  rounds from **R** to **S** and finally last  $r_{min}$  rounds from **S** to **R**.

Let  $q = t_b(r_{min} - 1)$  and  $n = 2q + 1$ . We refer to the nodes corresponding to the first  $r_{min}$  rounds from **S** to **R** as the first half denoted by  $\Gamma_i^{(1)}, 1 \leq i \leq 2q + 1$ , the nodes in the next  $r_{min}$  rounds from **R** to **S** as second half denoted by  $\Gamma_i^{(2)}, 1 \leq i \leq 2q + 1$  and the nodes in the last  $r_{min}$  rounds from **S** to **R** as third half denoted by  $\Gamma_i^{(3)}, 1 \leq i \leq 2q + 1$ . From Theorem 4.18,  $r_{min} = N - 2t_b + 1$ .

Protocol PRMT-Round-Mobile-Byzantine is same as protocol 3-Optimal-PRMT-Mobile-Byzantine, except that degree of each bi-variate polynomial is  $q$ . Moreover, **Phase**  $i, 1 \leq i \leq 3$  is executed in  $r_{min}$  rounds on  $\Gamma_j^{(i)}, 1 \leq j \leq 2q + 1$ . PRMT-Round-Mobile-Byzantine can be simulated on  $G$  following the explanation provided earlier for REL protocol. It is easy to see that Lemma 4.5 and Lemma 4.6 will hold for PRMT-Round-Mobile-Byzantine, with  $q$  in place of  $t_b$ . Protocol PRMT-Round-Mobile-Byzantine reliably sends  $n(q + 1)^2 = \Theta(n^3)$  field elements by communicating  $\mathcal{O}(n^3)$  field elements in  $3r_{min}$  rounds. The protocol is formally given in Fig. 4.8.

Figure 4.8: A  $3r_{min}$  Round Communication Optimal PRMT Protocol in Terms of Rounds

<b>Protocol PRMT-Round-Mobile-Byzantine</b>
<p>Let the sequence of <math>n(q+1)^2</math> field elements that <b>S</b> wishes to transmit be denoted by <math>m_{k,i,j}^{\mathbf{S}}</math>, <math>0 \leq i, j \leq q</math> and <math>1 \leq k \leq n</math>.</p>
<p><b>First <math>r_{min}</math> rounds: (S to R) executed over space time paths <math>\Gamma_i^{(1)}</math></b></p> <ol style="list-style-type: none"> <li>1. Using the <math>m_{k,i,j}^{\mathbf{S}}</math> values, <b>S</b> defines <math>n</math> bivariate polynomials <math>F_k^{\mathbf{S}}(x, y)</math>, <math>1 \leq k \leq n</math> as follows: <math display="block">F_k^{\mathbf{S}}(x, y) = \sum_{i=0, j=0}^{i=q, j=q} m_{k,i,j}^{\mathbf{S}} x^i y^j.</math> </li> <li>2. <b>S</b> evaluates each <math>F_k^{\mathbf{S}}(x, y)</math>, <math>1 \leq k \leq n</math> at <math>y = 1, \dots, n</math> to obtain total <math>n^2</math> polynomials denoted as <math>p_{k,i}^{\mathbf{S}}(x)</math>, <math>1 \leq k, i \leq n</math>, each of degree <math>q</math> where <math>p_{k,i}^{\mathbf{S}}(x) = F_k^{\mathbf{S}}(x, i)</math>.</li> <li>3. Over space time paths <math>\Gamma_i^{(1)}</math>, <math>1 \leq i \leq 2q+1</math>, <b>S</b> sends <math>p_{k,i}^{\mathbf{S}}(x)</math>, <math>1 \leq k \leq n</math> and the values <math>p_{k,j}^{\mathbf{S}}(i)</math>, denoted by <math>r_{k,j,i}^{\mathbf{S}}</math>, for <math>1 \leq k, j \leq n</math>.</li> </ol>
<p><b>Second <math>r_{min}</math> rounds: (R to S) executed over space time paths <math>\Gamma_i^{(2)}</math></b></p> <ol style="list-style-type: none"> <li>1. Let <b>R</b> receive over space time path <math>\Gamma_i^{(1)}</math>, <math>1 \leq i \leq n</math> the polynomials <math>p_{k,i}^{\mathbf{R}}(x)</math> and the values <math>r_{k,j,i}^{\mathbf{R}}</math>, <math>1 \leq k, j \leq n</math>.</li> <li>2. Using the received values, <b>R</b> constructs the conflict graphs <math>H_1, \dots, H_n</math>.</li> <li>3. <b>R</b> combines <math>H_k</math>, <math>1 \leq k \leq n</math> into a single conflict graph <math>H^{\mathbf{R}}</math> using Union Technique and forms the corresponding list of seven tuples <math>X</math> and reliably sends <math>X</math> to <b>S</b> by executing protocol REL over the space time paths <math>\Gamma_i^{(2)}</math>, <math>1 \leq i \leq 2q+1</math>.</li> </ol>
<p><b>Last <math>r_{min}</math> rounds: S to R executed over space time paths <math>\Gamma_i^{(3)}</math></b></p> <ol style="list-style-type: none"> <li>1. At the end of REL, <b>S</b> reliably receives the list <math>X</math> and identifies all faulty space time paths <math>\Gamma_i^{(1)}</math> over which <b>R</b> has received at least one faulty polynomial <math>p_{k,i}^{\mathbf{R}}(x) \neq p_{k,i}^{\mathbf{S}}(x)</math> during first <math>r_{min}</math> rounds.</li> <li>2. <b>S</b> adds all faulty paths to a list <math>L_{fault}</math>. Note that <math> L_{fault}  \leq q</math>. <b>S</b> then reliably sends <math>L_{fault}</math> to <b>R</b> by executing protocol REL protocol over the space time paths <math>\Gamma_i^{(3)}</math>, <math>1 \leq i \leq 2q+1</math> and terminates PRMT-Round-Mobile-Byzantine.</li> </ol>
<p><b>Message Recovery by R.</b></p> <ol style="list-style-type: none"> <li>1. <b>R</b> reliably receives <math>L_{fault}</math> at the end of REL and identifies all space time path <math>\Gamma_i^{(1)}</math> over which it has received at least one faulty polynomial <math>p_{k,i}^{\mathbf{R}}(x) \neq p_{k,i}^{\mathbf{S}}(x)</math> during the first <math>r_{min}</math> rounds (proof is similar to Lemma 4.5).</li> <li>2. <b>R</b> neglects all the polynomials <math>p_{k,i}^{\mathbf{R}}(x)</math>, <math>1 \leq k \leq n</math> received over each <math>\Gamma_i^{(1)} \in L_{fault}</math> during first <math>r_{min}</math> rounds.</li> <li>3. Using the remaining (at least <math>q+1</math>) <math>p_{k,i}^{\mathbf{R}}(x)</math>'s, <math>1 \leq k \leq n</math>, <b>R</b> correctly recovers the bivariate polynomials <math>F_k^{\mathbf{S}}(x, y)</math>'s, <math>1 \leq k \leq n</math> and hence the message and terminates PRMT-Round-Mobile-Byzantine.</li> </ol>

**Lemma 4.26** *Protocol PRMT-Round-Mobile-Byzantine is both round optimal as well as communication optimal.*

PROOF: The proof simply follows from the fact that protocol PRMT-Round-Mobile-Byzantine simulates the phase optimal and communication optimal three phase protocol 3-PRMT-Mobile-Byzantine and it takes  $r_{min}$  rounds to simulate a phase of protocol PRMT-Round-Mobile-Byzantine.  $\square$

In protocol PRMT-Round-Mobile-Byzantine, we have assumed that the adversary has roaming speed of  $\rho = 1$  and there are  $2t_b + 1$  node disjoint paths between **S** and **R**.

However, the protocol can be easily modified if there are more than  $2t_b + 1$  paths between  $\mathbf{S}$  and  $\mathbf{R}$ . This is because we can use **Algorithm-Round-Complexity** to compute  $r_{min}$ . Similarly, if  $\rho > 1$ , then also we can compute  $r_{min}^\rho$  and accordingly modify protocol PRMT-Round-Mobile-Byzantine.

## 4.6 Concluding Remarks and Open Problems

On the first look, a mobile adversary appears to be much more powerful and demanding than a static adversary with the same threshold. However the equivalence in terms of tolerability for these two kind of adversaries has been shown in [82]. In this chapter we have shown the equivalence in terms of designing optimal PRMT protocol. Specifically, we presented a three phase communication optimal PRMT protocol in undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{mobile}$ . Moreover, we have shown that our protocol is phase optimal, as well as require minimum connectivity. Our protocol takes three phases and sends a message of size  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements. Our protocol, along with Theorem 4.1 completely settles the issue of POSSIBILITY, FEASIBILITY and OPTIMALITY of PRMT in undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{mobile}$ . These results are summarized in Fig. 4.9.

Figure 4.9: Summary of the Results for PRMT in Undirected Synchronous Network Tolerating  $\mathcal{A}_{t_b}^{mobile}$

Number of Phases ( $r$ )	Connectivity Requirement ( $n$ )	Lower Bound on Communication Complexity	Upper Bound
$r \leq 2$	$n \geq 2t_b + 1$ Theorem 4.1	$\Omega\left(\frac{n\ell}{n-2t_b}\right)$ Theorem 4.2	Broadcast protocol: $n = 2t_b + 1$ , Communication complexity = $\mathcal{O}(n\ell)$
$r \geq 3$	$n \geq 2t_b + 1$ Theorem 4.1	$\Omega\left(\frac{n\ell}{n-t_b}\right)$ Theorem 4.3	Protocol 3-Optimal-PRMT-Mobile-Byzantine: $n = 2t_b + 1, \ell = \Theta(n^3)$ Communication complexity = $\mathcal{O}(\ell)$

From Fig. 4.9, we find that protocol 3-Optimal-PRMT-Mobile-Byzantine is communication optimal only if the message contains  $\ell = \Theta(n^3)$  field elements. This leads to the following open question:

**Open Problem 2** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + 1$  wires. Then does there exist a multiphase (more than two phase) PRMT protocol which reliably sends a message containing  $\ell$  field elements, by communicating  $\mathcal{O}(\ell)$  field elements, tolerating  $\mathcal{A}_{t_b}^{mobile}$ , for all values of  $\ell$  ?*

Our second major contribution in this chapter comes in terms of providing a generic method to compute the minimum number of rounds for PRMT tolerating a mobile adversary with arbitrary roaming speed. Though we have presented round optimal protocol for reliable communication for every tolerable adversary, we are able to show that the round optimal protocol are efficient only if the network is given as a collection of node disjoint paths. This leads to the following interesting open problem:

**Open Problem 3** *Given an arbitrary network and a  $t_b$ -active mobile adversary, with arbitrary roaming speed  $\rho$ , what is the minimum number of rounds required for reliable communication?*

Till now we have considered only Byzantine adversary, who corrupts the nodes in Byzantine fashion. In the next chapter, we consider a mixed adversary, who can corrupt nodes in a variety of ways.

## Chapter 5

# On Tradeoff Among Network Connectivity, Phase Complexity and Communication Complexity of PRMT Tolerating Static Mixed Adversary

In the last two chapters, we have studied PRMT tolerating only Byzantine corruption. However, in a typical large network, there can be various other type of corruption. For example, certain nodes may be strongly protected and few others may be moderately/weakly protected. An adversary may only be able to fail-stop a strongly protected node, while he may affect in a Byzantine fashion a weakly protected node. Thus, we may capture the abilities of an adversary in a more realistic manner using two parameters  $t_b$  and  $t_f$ , where  $t_b$  and  $t_f$  are the number of nodes corrupted in Byzantine and fail-stop respectively. Also it is better to grade different kinds of disruption done by adversary and consider them separately rather than treating every kind of fault as Byzantine fault because this is an “overkill” (a more detailed discussion on the last point will appear in the subsequent section of this chapter). So in this chapter, we study PRMT in *undirected synchronous network, tolerating threshold static mixed adversary*. We now give the formal specification of the network model used in this chapter.

### 5.1 Underlying Network Model and Adversary Settings

The underlying network model is similar as in Chapter 3. However, instead of a static Byzantine adversary  $\mathcal{A}_{t_b}^{static}$ , we assume the presence of a static mixed adversary  $\mathcal{A}_{(t_b, t_f)}^{static}$ . The adversary  $\mathcal{A}_{(t_b, t_f)}^{static}$  has *unbounded computing power* and controls *disjoint* sets of  $t_b$  and  $t_f$  nodes in Byzantine and fail-stop fashion<sup>1</sup> respectively. Thus, we assume that  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n$  parallel and synchronous bi-directional node disjoint paths/channels  $w_1, w_2, \dots, w_n$ , also called as *wires*. The adversary  $\mathcal{A}_{(t_b, t_f)}^{static}$ , having unbounded computing power can corrupt up to  $t_b$  and  $t_f$  wires in Byzantine and fail-stop fashion respectively. Moreover, as a worst case assumption, we assume that the wires that are under the control of the adversary in Byzantine and fail-stop fashion are mutually disjoint.

---

<sup>1</sup>See Section 2.1 for the definition of fail-stop corruption.

The characteristic of a Byzantine corrupted wire is already specified in Chapter 3. A wire which is controlled in a fail-stop fashion may fail to deliver any information, but if it delivers the information then it will be correct. Moreover, the adversary will have no idea about the information that has passed through a wire which is controlled in fail-stop fashion.

We assume that any PRMT protocol operates as a sequence of *phases*. The static mixed adversary  $\mathcal{A}_{(t_b, t_f)}^{static}$  controls the **same** set of  $t_b$  ( $t_f$ ) wires among  $n$  wires, in Byzantine (fail-stop) respectively, in different phases of any PRMT protocol. The static Byzantine adversary  $\mathcal{A}_{t_b}^{static}$  is a special type of  $\mathcal{A}_{(t_b, t_f)}^{static}$  with  $t_f = 0$ , who controls at most  $t_b$  wires in Byzantine fashion.

## 5.2 Existing Results for PRMT Tolerating $\mathcal{A}_{(t_b, t_f)}^{static}$

In [75], the author gave the following characterization for PRMT in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ .

**Theorem 5.1 ([75])** *Let  $r \geq 1$ . Then any  $r$ -phase PRMT protocol over an undirected synchronous network  $\mathcal{N}$  tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$  is possible iff  $\mathcal{N}$  is  $(2t_b + t_f + 1)$ -**(S, R)**-connected.*

PROOF (SKETCH): The necessity of the above condition follows from the following argument: Let  $\Pi$  be a PRMT protocol over  $\mathcal{N}$ , where there exists  $n = 2t_b + t_f$  wires between **S** and **R**. Now consider the following adversarial strategy of  $\mathcal{A}_{(t_b, t_f)}^{static}$ : the adversary blocks the communication over  $t_f$  wires. Now consider the network  $\mathcal{N}'$  that is induced by  $\mathcal{N}$  after deleting these  $t_f$  wires from  $\mathcal{N}$  (this can be interpreted as adversary  $\mathcal{A}_{(t_b, t_f)}^{static}$  blocking the communication over  $t_f$  wires). It follows that  $\mathcal{N}'$  is not a  $(2t_b + 1)$ -**(S, R)**-connected network. Evidently, if  $\Pi$  is a PRMT protocol in  $\mathcal{N}$  tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ , then  $\Pi'$  is a PRMT protocol in  $\mathcal{N}'$  tolerating  $\mathcal{A}_{t_b}^{static}$ , where  $\Pi'$  is the protocol  $\Pi$  restricted to the nodes in  $\mathcal{N}'$ . However, from Theorem 3.2, we know that  $\Pi'$  is non-existent. Thus  $\Pi$  is impossible too.

The sufficiency of the above condition is shown by the following protocol: Let **S** and **R** be connected by  $n = 2t_b + t_f + 1$  wires. **S** broadcasts the message  $m^{\mathbf{S}}$  through all the  $n$  wires. Out of the  $n$  wires, at most  $t_f$  wires may fail to deliver any information, while  $t_b$  wires may deliver incorrect information. However, there exists at least  $t_b + 1$  honest wires, which will deliver correct  $m^{\mathbf{S}}$ . Hence **R** recovers  $m^{\mathbf{S}}$  by taking majority among the received values.  $\square$

### 5.2.1 Justification to Study PRMT Tolerating $\mathcal{A}_{(t_b, t_f)}^{static}$

We now demonstrate that Theorem 5.1 shows more fault tolerance in comparison to Theorem 3.2. Let  $\mathcal{N}$  be a network where **S** and **R** are connected by  $n = 4$  wires. Then from Theorem 3.2, the maximum number of Byzantine faults that can be tolerated is one. However, from Theorem 5.1, it is possible to tolerate one fail-stop fault, in addition to one Byzantine fault. Thus Theorem 5.1 shows the availability of more fault tolerance in comparison to Theorem 3.2. If we would have modelled the fail-stop corruption as Byzantine corruption, then from Theorem 3.2, we would have required six wires, which is clearly more than what is actually required. This justifies the study of mixed adversary in the context of PRMT.

In the next section, we define the holy grail problem of PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ .

### 5.3 Holy Grail Problem of PRMT Tolerating $\mathcal{A}_{(t_b, t_f)}^{static}$

We now formulate the holy grail problem of PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ . PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$  is possible iff there exists  $n \geq 2t_b + t_f + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$  (see Theorem 5.1). Let us first consider the scenario where we work with minimal connectivity required for the existence of PRMT against  $\mathcal{A}_{(t_b, t_f)}^{static}$ ; viz  $n = 2t_b + t_f + 1$ . From the basic results of coding theory (see Theorem 5.5), any *single phase* PRMT protocol over  $n = 2t_b + t_f + 1$  wires has to communicate  $\Omega(n\ell)$  field elements to reliably send a message  $m^{\mathbf{S}}$ , containing  $\ell$  field elements. In fact, there exists a naive single phase PRMT protocol (namely the broadcast protocol) with a communication complexity of  $\mathcal{O}(n\ell)$ . So any PRMT protocol with communication complexity of less than  $n\ell$ , *must* run for several phases. Hence a natural and fundamental question here is the following:

*Given  $n = 2t_b + t_f + 1$ , a message  $m^{\mathbf{S}}$  of size  $\ell$  field elements and  $b < n\ell$ , what is the lower bound  $\Omega(r)$  on the phase complexity of any PRMT protocol, which sends  $m^{\mathbf{S}}$  with a total communication complexity of  $\mathcal{O}(b)$ ? Moreover, do we have such an  $\mathcal{O}(r)$  phase efficient PRMT protocol, whose total communication complexity is  $\mathcal{O}(b)$ ?*

It is clear that for any PRMT protocol, the communication complexity  $b$  should satisfy  $\ell \leq b \leq n\ell$ . This is because any PRMT protocol has to at least send the message. So  $\Omega(\ell)$  is a trivial lower bound on the communication complexity of any PRMT protocol. On the other hand, there exists a naive, single phase PRMT protocol with a communication complexity of  $\mathcal{O}(n\ell)$ . We may refer a PRMT protocol with communication complexity of  $\mathcal{O}(\ell)$ , as *communication optimal* PRMT protocol or PRMT protocol with *constant factor overhead*. Extending the previous question to this very interesting and important case, we may ask the following:

*When  $n = 2t_b + t_f + 1$ ,  $|m^{\mathbf{S}}| = \ell$  and  $b = \ell$ , what is the lower bound  $\Omega(r)$  on the phase complexity of any PRMT protocol, which sends  $m^{\mathbf{S}}$  with a total communication complexity of  $\mathcal{O}(b)$ ? Moreover, do we have such an  $\mathcal{O}(r)$  phase efficient PRMT protocol?*

Note that if such a protocol exists, it will be simultaneously optimal in connectivity, communication complexity and phase complexity.

So far, we have considered only minimally connected network. If we have higher connectivity, then again the required number of phases may be reduced. Specifically, when  $n \geq 2t_b + t_f + 1$  and  $b < n\ell$ , we ask for minimum  $r$  and a corresponding phase optimal protocol. Unifying all the above questions, we formulate the following most generic question, which is the *holy grail for PRMT problem*:

*Given an  $n$ -connected network ( $n \geq 2t_b + t_f + 1$ ) under the influence of  $\mathcal{A}_{(t_b, t_f)}^{static}$ , a message  $m^{\mathbf{S}}$  of size  $\ell$  field elements and a value  $b$ , where  $\ell \leq b < n\ell$ , what is the lower bound  $\Omega(r)$  on the phase complexity of any PRMT protocol, which sends  $m^{\mathbf{S}}$  with a total communication complexity of  $\mathcal{O}(b)$ ? Moreover, do we have such an  $\mathcal{O}(r)$  phase efficient PRMT protocol?*

To the best of our knowledge, no one has addressed the above all encompassing and unifying question.

## 5.4 Our Solution to the Holy Grail Problem

In this chapter, we completely resolve the holy grail question by deriving exact expression for lower bound for the phase complexity of PRMT protocols and also design a PRMT protocol whose phase complexity matches this bound asymptotically, thus proving that our bound is *asymptotically* tight. Our lower bound expression is first of its kind, which simultaneously captures the inherent relationship among all the parameters of any PRMT protocol, namely  $n, b$  and  $r$ . The existing lower bound(s) and the corresponding optimal protocols [28, 77] in the context of PRMT, try to optimize *only one* parameter out of  $n, b$  and  $r$ .

From our general result, we obtain several interesting corollaries on the inherent trade-off available between the three parameters, namely  $n, b$  and  $r$ . For example our general result imply that when  $t_b = 0, n = t_f + 1$  and  $b = \mathcal{O}(\ell)$ , then any PRMT protocol requires  $\Omega(\log(t_f))$  phases to send  $m^{\mathbf{S}}$ , where  $|m^{\mathbf{S}}| = \ell$ .

## 5.5 An Overview of Our Results in This Chapter

In this section, we give an overview of our results which are presented in this chapter. In summary we show the following in this chapter:

1. Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + t_f + 1$  wires. Then any single phase PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$  has to communicate  $\Omega\left(\frac{n\ell}{n - (2t_b + t_f)}\right)$  field elements to reliably send a message containing  $\ell$  field elements. Moreover this bound is tight. The lower bound is derived by showing equivalence of single phase PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$  and  $(t_b, t_f)$ -block error/erasure correcting codes.
2. If  $t_f \leq (n - t_b)$ , then there exists a three phase PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ , which can send  $m^{\mathbf{S}}$  containing  $\Theta(n^3)$  field elements by communicating  $\mathcal{O}(n^3)$  field elements.
3. Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + t_f + 1$  wires such that  $t_b > 0$  and  $t_f > (n - t_b)$ . Let  $m^{\mathbf{S}}$  be a message of size  $\ell$  field elements, which  $\mathbf{S}$  wants to reliably send to  $\mathbf{R}$ , tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ , such that the total communication complexity of the protocol is  $\mathcal{O}(b)$ , where  $\ell \leq b < n\ell$ . Then any PRMT protocol to send  $m^{\mathbf{S}}$  must run for  $\Omega\left(\frac{\log(\frac{t_f}{n - t_b})}{\log(\frac{cb}{\ell})}\right) = \Omega(\mathcal{D})$  phases, where  $c > 1$  is a positive constant.

We show that this bound on phase complexity is asymptotically tight. That is, we design an  $\mathcal{O}(\mathcal{D})$  phase PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ , where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n \geq 2t_b + t_f + 1$  wires. The protocol reliably sends a message  $m^{\mathbf{S}}$  containing  $\ell = n^3$  field elements by communicating  $\mathcal{O}(b)$  field elements, where  $\ell \leq b < n\ell$ .

As important corollaries of the above lower bound, we also show the following:

- (a) Any PRMT protocol with constant factor overhead over  $n = t_f + 1$  wires, influenced by  $\mathcal{A}_{t_f}^{static}$  (i.e.,  $t_b = 0$ ), must run for  $\Omega(\log(t_f))$  phases.
- (b) Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $2t_b + t_f + 1$  wires, such that  $t_b, t_f > 0$  and  $t_f > (n - t_b)$ . Then any PRMT protocol with constant factor overhead tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ , must run for  $\Omega(\log(\frac{t_f}{t_b}))$  phases.

**Remark 5.2 (Surprising Factor)** *As mentioned above, if  $t_b = 0, n = t_f + 1$  and  $b = \mathcal{O}(\ell)$ , then any PRMT protocol requires  $\Omega(\log(t_f))$  phases to send  $m^{\mathbf{S}}$  containing  $\ell$  field elements against  $\mathcal{A}_{t_f}^{\text{static}}$ . This is surprising because from Theorem 3.16, against only static Byzantine adversary  $\mathcal{A}_{t_b}^{\text{static}}$  (i.e.,  $t_f = 0$ ), there exists a constant (three) phase PRMT protocol, which achieves reliability with constant factor overhead. These two results seem to be counter intuitive, since a Byzantine adversary is more powerful than fail-stop adversary (a Byzantine adversary can maliciously change the information over a wire, where as fail-stop adversary can only block the communication over a wire). However, we informally explain the subtle but simple reason behind this seemingly paradoxical result (a formal argument is given in the subsequent section).*

*In a minimally connected network tolerating only Byzantine adversary, we have  $n = 2t_b + 1$  wires. So the number of corrupted wires is less than half of the total number of wires. This allow us to do error detection/correction in three phases, resulting in a three phase PRMT protocol with a communication complexity of  $\mathcal{O}(\ell)$  (for details, see protocol 3-Optimal-PRMT-Static-Byzantine in Fig. 3.4). However, in a minimally connected network tolerating only fail-stop adversary, we have only  $n = t_f + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ , of which there is only one un-corrupted wire. It is this reduced connectivity (in comparison to the case of a Byzantine adversary) that hinders us to design a constant phase PRMT protocol against fail-stop adversary in a minimally connected network, with communication complexity of  $\mathcal{O}(\ell)$ .*

In the next section, we derive the lower bound on the communication complexity of single phase PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{static}}$ .

## 5.6 Bounds for Single Phase PRMT Tolerating $\mathcal{A}_{(t_b, t_f)}^{\text{static}}$

In this section, we show the equivalence between a single phase PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{static}}$  and  $(t_b, t_f)$ -block error-erasure correcting code. Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $N \geq 2t_b + T + 1$  wires,  $w_1, \dots, w_N$ , of which at most  $t_b$  and  $T$  wires can be Byzantine and fail-stop corrupted, such that  $T \leq t_f$ . We design a single phase PRMT protocol called 1-PRMT-Mixed, which allows  $\mathbf{S}$  to reliably send a message  $m^{\mathbf{S}}$  containing  $\ell \geq 1$  field elements to  $\mathbf{R}$ . The protocol is based on the properties of RS codes and is presented in Fig. 5.1.

We now prove the properties of protocol 1-PRMT-Mixed.

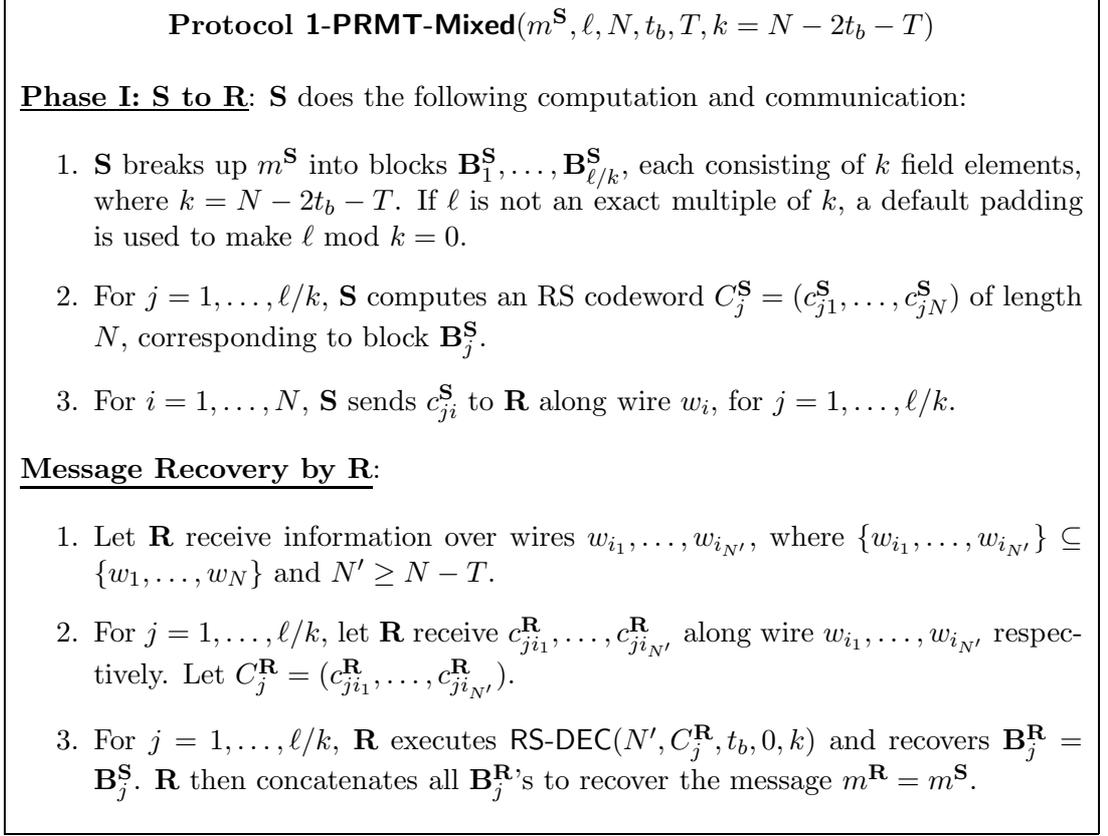
**Lemma 5.3 (Correctness)** *Protocol 1-PRMT-Mixed correctly delivers  $m^{\mathbf{S}}$ .*

PROOF: In order to show that  $\mathbf{R}$  will correctly receive  $m^{\mathbf{S}}$ , we show that  $\mathbf{R}$  will recover each  $\mathbf{B}_j^{\mathbf{S}}$  correctly. In the protocol, each  $\mathbf{B}_j^{\mathbf{S}}$  is of size  $k = N - 2t_b - T$  and is RS encoded into a codeword of length  $N \geq 2t_b + T + 1$ . Corresponding to each  $\mathbf{B}_j^{\mathbf{S}}$ ,  $\mathbf{R}$  receives a vector  $C_j^{\mathbf{R}}$  of size  $N'$  where  $N' \geq N - T$  and this vector  $C_j^{\mathbf{R}}$  differs from the original codeword  $C_j^{\mathbf{S}}$  in at most  $t_b$  locations. So by putting  $N' \geq N - T, k = N - 2t_b - T, c = t_b$  and  $d = 0$  in the inequality of Theorem 2.19, we find that  $\mathbf{R}$  will be able to correct all the  $t_b$  errors in  $C_j^{\mathbf{R}}$  by applying RS-DEC to  $C_j^{\mathbf{R}}$ . Thus  $\mathbf{R}$  correctly recovers  $\mathbf{B}_j^{\mathbf{S}}$ .  $\square$

**Lemma 5.4 (Communication Complexity)** *The communication complexity of protocol 1-PRMT-Mixed is  $\mathcal{O}\left(\frac{N\ell}{N-2t_b-T}\right)$ .*

PROOF: Corresponding to each block of size  $k$ ,  $\mathbf{S}$  sends an RS codeword of length  $N$ . So communication complexity of the protocol is  $\mathcal{O}\left(\frac{N\ell}{k}\right) = \mathcal{O}\left(\frac{N\ell}{N-2t_b-T}\right)$ .  $\square$

Figure 5.1: Single Phase Reliable Message Transmission Tolerating Mixed Adversary



Putting  $T = t_f$ , the maximum value of  $T$  in protocol 1-PRMT-Mixed we find that given a  $(t_b, t_f)$ -block error-erasure correcting code (for example RS code), whose maximum attainable efficiency is bounded by Singleton Bound, we can design a single phase PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{static}}$ . The reverse process is equally valid - given a single phase PRMT protocol against  $\mathcal{A}_{(t_b, t_f)}^{\text{static}}$ , we can convert it into a  $(t_b, t_f)$ -block error-erasure correcting code, whose efficiency is bounded by Singleton Bound. Thus, the maximum attainable efficiency for any single phase PRMT protocol is also subject to the Singleton Bound. Thus we have the following theorem.

**Theorem 5.5** *Let **S** and **R** be connected by  $n \geq 2t_b + t_f + 1$  wires. Then any single phase PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{static}}$  must communicate  $\Omega\left(\frac{n\ell}{n-(2t_b+t_f)}\right)$  field elements to reliably send  $m^{\mathbf{S}}$ , where  $|m^{\mathbf{S}}| = \ell$ . Moreover, the bound is tight.*

**Remark 5.6** *The conversion from single phase PRMT protocol to an error-erasure correcting code is straightforward if the messages sent along each wire in the protocol are of same length. Suppose however, there exists a protocol  $\Pi$  that does not have this symmetry property and beats the Singleton bound. Then consider the protocol  $\Pi'$  which consists of  $n$  sequential executions of protocol  $\Pi$  with the identities of the wires being “rotated” by a distance of  $i$  in the  $i^{\text{th}}$  execution. Clearly this protocol achieves the symmetry property by “spreading the load”; further its message expansion factor is equal to that of  $\Pi$ . It therefore beats the Singleton bounds as well, which is a contradiction. Thus without any loss of generality, we assume that the messages sent along each wire in  $\Pi$  are of same length.*

### 5.6.1 Increasing the Throughput of Protocol 1-PRMT-Mixed

Protocol 1-PRMT-Mixed has another important property. Consider the following scenario: Let  $\mathbf{S}$  know that  $\mathbf{R}$  has the knowledge of the exact identity of  $\alpha \leq t_b$  wires that are Byzantine corrupted. However  $\mathbf{S}$  does not know the exact identity of those  $\alpha$  wires. If this is the case, then the following theorem holds:

**Theorem 5.7 (Increased Throughput in 1-PRMT-Mixed)** *Suppose  $\mathbf{S}$  knows that  $\mathbf{R}$  has the knowledge of the exact identity of  $\alpha \leq t_b$  wires that are Byzantine corrupted. Then in protocol 1-PRMT-Mixed,  $\mathbf{S}$  can reliably send  $m^{\mathbf{S}}$  using block size  $k = (N - 2t_b - T) + \alpha$ . That is,  $\mathbf{S}$  can send additional  $\alpha$  field elements, than what is allowed by Singleton bound. Moreover, the communication complexity of the protocol will be  $\mathcal{O}\left(\frac{N\ell}{(N-2t_b-T)+\alpha}\right)$*

PROOF: Since  $\mathbf{R}$  is aware of the exact identity of  $\alpha$  Byzantine corrupted wires,  $\mathbf{R}$  can simply ignore the values received over these wires. So the length of each received vector  $C_j^{\mathbf{R}}$  will be  $N'$ , where  $N' \geq N - T - \alpha$ . Moreover  $C_j^{\mathbf{R}}$  will now differ from original codeword  $C_j^{\mathbf{S}}$  in at most  $t_b - \alpha$  locations. So by putting  $N' \geq N - T - \alpha, k = (N - 2t_b - T) + \alpha, c = t_b - \alpha$  and  $d = 0$  in the inequality of Theorem 2.19, we find that by applying RS-DEC to  $C_j^{\mathbf{R}}$ ,  $\mathbf{R}$  will be able to correct all the  $t_b - \alpha$  errors in  $C_j^{\mathbf{R}}$  and hence correctly recover  $\mathbf{B}_j^{\mathbf{R}}$ .

Since  $k = (N - 2t_b - T) + \alpha$ , the communication complexity of the protocol will be  $\mathcal{O}\left(\frac{N\ell}{(N-2t_b-T)+\alpha}\right)$ .  $\square$

### 5.6.2 PRMT Based on Both Error Correction and Detection Capability

Protocol 1-PRMT-Mixed uses only the error correcting capability of underlying error-erasure correcting code. We now present a two phase protocol called 2-SP-REL-Mixed (based on RS code which is a specific instance of error-erasure correcting code) that possesses both error correction and error detection capabilities of the underlying error-erasure correcting code. This is used later in designing our phase optimal PRMT protocol. In 2-SP-REL-Mixed,  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $N \geq 2t_b + T + 1$  wires, of which at most  $t_b$  and  $T$  wires can be Byzantine and fail-stop corrupted. Moreover,  $T \leq t_f$ .

2-SP-REL-Mixed is based on the following principle:  $\mathbf{S}$  and  $\mathbf{R}$  guesses that adversary will fail-stop at most  $T - k_f$  wires and Byzantine corrupt at most  $\frac{t_b}{2}$  wires, where  $0 \leq k_f \leq T$ . If the adversary indeed does so, then  $\mathbf{S}$  can reliably send  $\frac{t_b}{2} + k_f$  extra field elements (in addition to what is permitted by Singleton Bound) in a single phase to  $\mathbf{R}$ . However if adversary either fail-stops  $T - k_f + 1$  (or more) wires or Byzantine corrupts more than  $\frac{t_b}{2}$  wires, then  $\mathbf{R}$  will fail to recover anything. But  $\mathbf{R}$  either comes to know the identity of at least  $T - k_f + 1$  fail-stop wires or detects more than  $\frac{t_b}{2}$  Byzantine faults. In the later case,  $\mathbf{R}$  can broadcast his findings to  $\mathbf{S}$ , who after local verification can identify more than  $\frac{t_b}{2}$  Byzantine corrupted wires. The protocol is presented in Fig. 5.2.

We now prove the properties of protocol 2-SP-REL-Mixed.

**Lemma 5.8 (Correctness)** *In 2-SP-REL-Mixed:*

1. *If more than  $T - k_f$  fail-stop errors occur during Phase I then  $\mathbf{R}$  will not be able to recover  $m^{\mathbf{S}}$ . However, in this case  $\mathbf{R}$  will be able to know the identity of more*

Figure 5.2: Protocol 2-SP-REL-Mixed

**Protocol 2-SP-REL-Mixed** ( $m^{\mathbf{S}}, N, t_b, T, k_f$ )

**Phase I: S to R**

1. **S** performs the same computation and communication as done in protocol 1-PRMT-Mixed, except that now **S** divides  $m^{\mathbf{S}}$  into blocks  $\mathbf{B}_1^{\mathbf{S}}, \dots, \mathbf{B}_z^{\mathbf{S}}$ , each consisting of  $k$  field elements, where  $k = (N - 2t_b - T) + \frac{t_b}{2} + k_f$ .

**Phase II: R to S**

1. Let **R** receive information over  $N'$  wires, of which at most  $t_b$  could be corrupted. Thus **R** receives  $N'$  components, corresponding to each of the  $z$  code-words.
2. IF  $N' < N - T + k_f$  then **R** broadcasts to **S**, "ERROR1" signal along with the count of the number of wires and their identity over which **R** has not received any information and terminates. This case implies that more than  $T - k_f$  fail-stop errors have occurred during the transmission by **S**.
3. IF  $N \geq N - T + k_f$  then **R** applies RS-DEC( $N', C_i^{\mathbf{R}}, \frac{t_b}{2}, \frac{t_b}{2}, k$ ), where for  $i = 1, \dots, z$ ,  $C_i^{\mathbf{R}}$  is the vector of length  $N'$ , received corresponding to the block  $\mathbf{B}_i^{\mathbf{S}}$ . Now there are two possible cases:
  - (a) If after correcting  $\frac{t_b}{2}$  errors, the decoding algorithm does not detect additional faults in any of the  $z$  received vectors, then **R** correctly recovers  $\mathbf{B}_i^{\mathbf{S}}$ ,  $1 \leq i \leq z$ . **R** then concatenates them to recover  $m^{\mathbf{S}}$ , broadcasts "SUCCESS" signal to **S** and terminates the protocol. This case implies that at most  $T - k_f$  fail-stop errors and at most  $\frac{t_b}{2}$  Byzantine errors have occurred during transmission by **S**.
  - (b) IF  $\exists e \in \{1, 2, \dots, z\}$ , such that after correcting  $\frac{t_b}{2}$  errors, the decoding algorithm detects additional faults (at most  $\frac{t_b}{2}$ ) in the  $e^{th}$  received vector  $C_e^{\mathbf{R}}$ , then **R** broadcasts  $C_e^{\mathbf{R}}$ , along with index  $e$  and "ERROR2" signal to **S**. If there are several such  $e$ 's, then **R** randomly selects one. **R** also broadcasts the identity of wires which failed to deliver any information and terminates. This case implies that at most  $T - k_f$  fail-stop errors and more than  $\frac{t_b}{2}$  Byzantine errors have occurred during transmission by **S**.

**Local Computation by S:**

1. IF **S** receives "SUCCESS" signal, then it does nothing.
2. IF **S** receives "ERROR1" signal, then **S** comes to know the identity of  $N - N'$  fail-stop corrupted wires.
3. IF **S** receives "ERROR2" signal, along with  $C_e^{\mathbf{R}}$  then after comparing it with original codeword  $C_e^{\mathbf{S}}$ , **S** identifies more than  $\frac{t_b}{2}$  wires which were Byzantine corrupted during **Phase I**.

than  $T - k_f$  fail-stop corrupted wires, which  $\mathbf{S}$  will also come to know at the end of **Phase II**.

2. If at most  $T - k_f$  fail-stop and  $\frac{t_b}{2}$  Byzantine errors occur during **Phase I**, then  $\mathbf{R}$  will be able to recover  $m^{\mathbf{S}}$ .
3. If at most  $T - k_f$  fail-stop errors and more than  $\frac{t_b}{2}$  Byzantine errors occur during **Phase I**, then  $\mathbf{R}$  will fail to recover  $m^{\mathbf{S}}$ . However, in this case  $\mathbf{R}$  will detect that more than  $\frac{t_b}{2}$  Byzantine errors have occurred during **Phase I**. Moreover, at the end of **Phase II**,  $\mathbf{S}$  will come to know the identity of more than  $\frac{t_b}{2}$  wires which were Byzantine corrupted during **Phase I**.

PROOF: We consider the following three cases:

1. **More than  $T - k_f$  wires get fail-stop corrupted during first phase:** In this case, irrespective of the number of Byzantine errors,  $\mathbf{R}$  will come to know the exact identity of more than  $T - k_f$  wires which are fail-stop corrupted and broadcasts their identity to  $\mathbf{S}$ . So  $\mathbf{S}$  will also come to know the identity of these fail-stop corrupted wires. Notice that in this case  $\mathbf{R}$  will fail to recover  $m^{\mathbf{S}}$ .
2. **At most  $T - k_f$  and  $\frac{t_b}{2}$  wires get fail-stop and Byzantine corrupted respectively during Phase I:** We consider the worst case, where exactly  $T - k_f$  wires get fail-stop corrupted during **Phase I**. Thus,  $\mathbf{R}$  will receive a vector  $C_i^{\mathbf{R}}$  of length  $N'$ , corresponding to each block  $\mathbf{B}_i^{\mathbf{S}}$ , where  $N' = N - (T - k_f)$ . Moreover, each  $C_i^{\mathbf{R}}$  will differ from  $C_i^{\mathbf{S}}$  in at most  $\frac{t_b}{2}$  locations. Notice that each  $\mathbf{B}_i^{\mathbf{S}}$  is RS encoded using a polynomial of degree  $k - 1 = (N - 2t_b - T) + \frac{t_b}{2} + k_f - 1$ . Substituting the value of  $N'$  and  $k$  in Theorem 2.19, we find that RS-DEC can correct  $c = \frac{t_b}{2}$  and detect additional  $d = \frac{t_b}{2}$  Byzantine errors in each received vector. Since the number of Byzantine errors in each received vector is at most  $\frac{t_b}{2}$ , the decoding algorithm will correct them (and does not detect any additional error) and recover each  $\mathbf{B}_i^{\mathbf{S}}$  (and hence  $m^{\mathbf{S}}$ ) correctly.
3. **At most  $T - k_f$  wires get fail-stop corrupted but more than  $\frac{t_b}{2}$  wires get Byzantine corrupted during Phase I:** Suppose more than  $\frac{t_b}{2}$  Byzantine errors occur during the transmission of  $e^{th}$  codeword  $C_e^{\mathbf{S}}$ , where  $e \in \{1, 2, \dots, z\}$ . In this case, from the above argument, the decoding algorithm will correct  $\frac{t_b}{2}$  errors and will detect additional errors (at most  $\frac{t_b}{2}$ ) in the  $e^{th}$  received vector  $C_e^{\mathbf{R}}$ . So  $\mathbf{R}$  will come to know that more than  $\frac{t_b}{2}$  Byzantine errors occurred during the transmission of  $C_e^{\mathbf{S}}$ . So  $\mathbf{R}$  broadcasts  $C_e^{\mathbf{R}}$  to  $\mathbf{S}$ , who after locally comparing it with the original  $e^{th}$  codeword  $C_e^{\mathbf{S}}$  finds the identity of more than  $\frac{t_b}{2}$  Byzantine corrupted wires. Notice that in this case,  $\mathbf{R}$  will fail to recover the block  $\mathbf{B}_e^{\mathbf{S}}$  and hence the message  $m^{\mathbf{S}}$ .  $\square$

**Lemma 5.9 (Communication Complexity)** *The communication complexity of protocol 2-SP-REL-Mixed is  $\mathcal{O}\left(\frac{|m^{\mathbf{S}}|N}{(N-2t_b-T)+\frac{t_b}{2}+k_f}\right) + \mathcal{O}(N^2)$ .*

PROOF: During **Phase I**,  $\mathbf{S}$  sends an RS codeword of length  $N$  for each sub-block of  $m^{\mathbf{S}}$ , where the size of each sub-block is  $(N - 2t_b - T) + \frac{t_b}{2} + k_f$ . This incurs a communication cost of  $\mathcal{O}\left(\frac{|m^{\mathbf{S}}|N}{(N-2t_b-T)+\frac{t_b}{2}+k_f}\right)$  field elements. During **Phase II**,  $\mathbf{R}$  broadcasts a signal and in the worst case may also broadcast a vector of length  $N$ . This incurs

a communication cost of  $\mathcal{O}(N^2)$ . So total communication cost of protocol 2-SP-REL-Mixed is  $\mathcal{O}\left(\frac{|m^{\mathbf{S}}|N}{(N-2t_b-T)+\frac{t_b}{2}+k_f}\right) + \mathcal{O}(N^2)$ .  $\square$

2-SP-REL-Mixed brings to the fore an important property (given in Corollary 5.9.1) which holds for any single phase PRMT protocol. This property will be later used to solve the holy grail problem of PRMT in the subsequent section.

**Corollary 5.9.1** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $N \geq 2t_b + T + 1$  wires, of which at most  $t_b$  and  $T$  wires can be Byzantine and fail-stop corrupted, such that  $T \leq t_f$ . Suppose  $\mathbf{S}$  wants to reliably send  $m^{\mathbf{S}}$  to  $\mathbf{R}$  where  $|m^{\mathbf{S}}| = \ell$ . If  $\mathbf{S}$  in advance knows that adversary will not do any Byzantine corruption and will only fail-stop at most  $T - k_f$  wires, where  $0 \leq k_f \leq T$ , then  $\mathbf{S}$  has to communicate at least  $\frac{N\ell}{(N-T+k_f)}$  field elements to reliably send  $m^{\mathbf{S}}$ . Moreover the minimum number of wires that the adversary needs to fail-stop in order that  $\mathbf{R}$  does not recover  $m^{\mathbf{S}}$  is  $T - k_f + 1$ . Thus if  $\mathbf{S}$  does not know in advance the number of fail-stop corruptions which are going to be done by the adversary, then  $\frac{N\ell}{N-T+k_f}$  is a trivial lower bound on the number of field elements to be sent by  $\mathbf{S}$ , so that either  $\mathbf{R}$  recovers  $m^{\mathbf{S}}$  (if no Byzantine corruption and at most  $T - k_f$  fail-stop corruptions occur) or  $\mathbf{R}$  comes to know the identity of at least  $T - k_f + 1$  fail-stop corrupted wires (if more than  $T - k_f$  fail-stop corruptions occur). The above expression can also be viewed as  $\ell \times \frac{X+T}{X+k_f}$ , where  $X = N - T$ .*

This finishes our discussion on single phase PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ . In the next section, we begin our discussion on the holy grail problem.

## 5.7 Answer to the Holy Grail Problem of PRMT

We now derive a nontrivial lower bound on phase complexity for any PRMT protocol which transmits  $\ell$  field elements by communicating  $\mathcal{O}(b)$  field elements against  $\mathcal{A}_{(t_b, t_f)}^{static}$ , where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n \geq 2t_b + t_f + 1$  wires and  $\ell \leq b < n\ell$ . This would completely resolve the holy grail problem for PRMT. Notice that we are considering  $n \geq 2t_b + t_f + 1$  because from Theorem 5.1, for the existence of any PRMT protocol against  $\mathcal{A}_{(t_b, t_f)}^{static}$ , there should exist  $n \geq 2t_b + t_f + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ . Similarly, we are interested in the case where  $\ell \leq b < n\ell$ . This is because any PRMT protocol has a non-trivial lower bound of  $\Omega(\ell)$  on its communication complexity. On the other hand,  $\mathbf{S}$  can trivially send  $m^{\mathbf{S}}$  in a single phase by broadcasting it over the  $n$  wires. This would require a communication cost of  $\mathcal{O}(n\ell)$ .

Recall that according to the definition of PRMT,  $\mathbf{R}$  should correctly output the message without any error. We assume that the protocol specification is public and adversary is also aware of the steps of the protocol. Accordingly, the adversary device his strategy. However, adversary has no access to the internal random coin tosses of  $\mathbf{S}$  and  $\mathbf{R}$ . Without loss of generality, we assume that during each phase of the protocol, the information sent over each wire is of the same length. Otherwise, using a similar argument as given in Remark 5.6, we can show that any PRMT protocol which sends un-equal sized information over each wire in each phase, does no better than a protocol which sends equal sized information over each wire in each phase.

**Theorem 5.10** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + t_f + 1$  wires such that  $t_f > 0$  and  $t_f > (n - t_f)$ . Then any PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ , must run for*

$\Omega\left(\frac{\log\left(\frac{t_f}{n-t_f}\right)}{\log\left(\frac{cb}{\ell}\right)}\right) = \Omega(\mathcal{D})$  phases for transmitting  $m^{\mathbf{S}}$  with a communication complexity of  $\mathcal{O}(b)$  field elements, where  $|m^{\mathbf{S}}| = \ell$ ,  $\ell \leq b < n\ell$  and  $c > 1$  is a positive constant.

**Remark 5.11** *The lower bound of  $\Omega(\mathcal{D})$  phases does not hold good if  $t_f = 0$ . If  $t_f = 0$ , then  $n \geq 2t_b + 1$ . In this case, there exist a three phase PRMT protocol (protocol 3-Optimal-PRMT-Static-Byzantine, given in Fig. 3.4) which achieves reliability with constant factor overhead. The lower bound also does not hold good if  $t_f \leq (n - t_f)$ . If  $t_f \leq (n - t_f)$ , then we can send  $m^{\mathbf{S}}$  by communicating  $\mathcal{O}(\ell)$  field elements in constant phases. A three phase PRMT protocol for achieving this task is given in section 5.8.1 (see Corollary 5.19.1).*

PROOF (of the theorem:) We present an adversarial behavior, against which no PRMT protocol can send  $m^{\mathbf{S}}$  with a communication complexity of  $\mathcal{O}(b)$  in less than  $\mathcal{D}$  phases. The adversary strategy is as follows: the adversary does no Byzantine corruption throughout the protocol. Any lower bound derived with this assumption is surely a lower bound when the adversary **indeed does** Byzantine corruption. So, here the adversary does only fail-stop corruption. Once a wire (corrupted in fail-stop fashion) fails to deliver information, it is marked as faulty wire and can be removed from the set of *current* (currently used) wires. Therefore, at the beginning of any protocol, number of current wires is  $n$  and number of *undisclosed* fail-stop wires is  $t_f$ . Whenever a wire (fail-stop corrupted) stops the communication, it reveals its corrupted status and thus number of *undisclosed* fail-stop wires reduces. Let the number of *undisclosed* fail-stop wires after  $i^{\text{th}}$  disclosure be denoted by  $L_i$ . Initially  $L_0 = t_f$ . Informally, the adversarial strategy is as follows: During each phase, adversary checks how much portion of  $m^{\mathbf{S}}$ ,  $\mathbf{S}$  is trying to send to  $\mathbf{R}$ . This he can find out from the protocol specification. If the size of the portion that  $\mathbf{S}$  tries to send during a particular phase is more than a “specific” limit, then the adversary does the minimum number of fail-stop corruptions, so that  $\mathbf{R}$  can recover only a specific “sub-portion” of the portion of  $m^{\mathbf{S}}$  sent by  $\mathbf{S}$  in that phase. Else, the adversary does no fail-stop corruption. Specifically, after the  $i^{\text{th}}$  disclosure of fail-stop corrupted wires, the adversary does the following:

- If  $\mathbf{S}$  tries to send a portion of size  $q \leq \frac{\ell \log\left(\frac{cb}{\ell}\right) \log(t_f)}{\log(L_i) \log\left(\frac{t_f}{n-t_f}\right)} = \frac{\ell \log(t_f)}{\mathcal{D} \log(L_i)}$  then adversary does nothing.

- If  $\mathbf{S}$  tries to send a portion of size more than  $\frac{\ell \log(t_f)}{\mathcal{D} \log(L_i)}$  then the adversary tries to fail-stop the minimum number of wires so that  $\mathbf{R}$  can recover only  $\frac{\ell \log(t_f)}{\mathcal{D} \log(L_i)}$  sub-portion of the total portion that has been sent by  $\mathbf{S}$ . If this is not possible then adversary fail-stops in such a way that the number of undisclosed fail-stop wires reduces to  $t_f^\epsilon$ , where  $\epsilon$  is a fixed positive fraction (which will be fixed shortly). Moreover, the adversary does no corruption at all for the rest of the protocol execution.

In the sequel, we consider three possible cases and prove that in each case, the number of phases required by the protocol is indeed  $\Omega(\mathcal{D})$  and also show that such an adversarial act is mountable against any protocol. During the computation of lower bound, we use logarithm to the base  $e$ .

**Claim 5.12** *If the adversary does no fail-stop corruption throughout the protocol, then the protocol terminates in  $\Omega(\mathcal{D})$  phases.*

PROOF: The reason why adversary remained inactive throughout the protocol is that  $\mathbf{S}$  never tried to transmit more than  $\frac{\ell \log(t_f)}{\mathcal{D} \log(t_f)}$  field elements from  $m^{\mathbf{S}}$  in any phase. Hence, denoting the maximum message size as  $q$  that has been sent in any phase, we get,  $q \leq \frac{\ell \log(t_f)}{\mathcal{D} \log(L_0)} \leq \frac{\ell \log(t_f)}{\mathcal{D} \log(t_f)} = \frac{\ell}{\mathcal{D}}$  because  $L_0 = t_f$ . So the minimum number of phases  $r$ , required to send  $\ell$  field elements is given by  $r \geq \frac{\ell}{q} = \mathcal{D} = \Omega(\mathcal{D})$ .  $\square$

**Claim 5.13** *If the adversary fail-stops in such a way that the number of undisclosed fail-stop wires at the end of the protocol remain strictly greater than  $t_f^\epsilon$ , then the protocol runs for  $\Omega(\mathcal{D})$  phases.*

PROOF: Clearly, in this case the maximum message size sent by  $\mathbf{S}$  in any phase is given by  $q \leq \frac{\ell \log(t_f)}{\mathcal{D} \log(t_f^\epsilon)} = \frac{\ell \log(t_f)}{\mathcal{D} \epsilon \log(t_f)} = \frac{\ell}{\epsilon \mathcal{D}}$ . So even if at most  $q$  field elements are communicated in each phase, the protocol takes  $r \geq \frac{\ell}{q} = \epsilon \mathcal{D} = \Omega(\mathcal{D})$  phases to send  $\ell$  field elements. Hence the claim.  $\square$

**Claim 5.14** *If the adversary fail-stops in such a manner that the number of undisclosed fail-stop wires become less than or equal to  $t_f^\epsilon$ , then also the protocol takes  $\Omega(\mathcal{D})$  phases.*

PROOF: So in this case the number of undisclosed fail-stop wires reduces down to less than or equal to  $t_f^\epsilon$ . Suppose the adversary used  $k$  phases denoted as  $Ph_1, Ph_2, \dots, Ph_k$  to reduce the number of undisclosed fail-stop wires from  $t_f$  to  $t_f^\epsilon$ . We will show that  $k$ , the number of phases in which fail-stop corruption occurs is  $\Omega(\mathcal{D})$ , excluding other phases where adversary does nothing. Let the number of undisclosed fail-stop wires after  $Ph_i$  reduces from  $L_{i-1}$  to  $L_i$ . So recording the count of undisclosed fail-stop wires after every disclosure (of adversary) starting with the initial count of  $L_0 = t_f$ , we get a decreasing sequence  $t_f, L_1, L_2, \dots, L_{k-1}, t_f^\epsilon$ . Let  $\alpha_1, \alpha_2, \dots, \alpha_k$  be the number of field elements communicated by  $\mathbf{S}$  in the corresponding phases. In phase  $Ph_i$ ,  $\mathbf{S}$  must have tried to send more than  $\frac{\ell \log(t_f)}{\mathcal{D} \log(L_{i-1})}$  field elements of  $m^{\mathbf{S}}$  and the adversary exposed the minimum number of fail-stop corrupted wires (hence after  $Ph_i$  reducing number of undisclosed fail-stop wires from  $L_{i-1}$  to  $L_i$ ) such that only  $\frac{\ell \log(t_f)}{\mathcal{D} \log(L_{i-1})}$  field elements of the total message is recoverable by  $\mathbf{R}$ . Then by Corollary 5.9.1, the number of field elements  $\alpha_i, 1 \leq i \leq k$  transmitted in  $Ph_i$  is given by  $\alpha_i \geq \frac{\ell \log(t_f)}{\mathcal{D} \log(L_{i-1})} \times \frac{X+L_{i-1}}{X+L_i}$ , where  $X = n - t_f$ . This is so because during phase  $Ph_i$ , the number of current wires is  $X + L_{i-1}$ , the number of unknown errors (fail-stop) is  $L_{i-1}$  and  $\mathbf{S}$  tried to sent at least  $\frac{\ell \log(t_f)}{\mathcal{D} \log(L_{i-1})}$  field elements. The above mentioned attack by the adversary is mountable since adversary is aware of  $\alpha_i$  (adversary knows the protocol specification) and can solve  $\frac{\ell \log(t_f)}{\mathcal{D} \log(L_{i-1})} \times \frac{X+L_{i-1}}{X+x} = \alpha_i$  for  $x$  and accordingly blocks only  $L_{i-1} - x$  wires, so that  $\mathbf{R}$  recovers only  $\frac{\ell \log(t_f)}{\mathcal{D} \log(L_{i-1})}$  field elements and  $L_{i-1}$  reduces to  $L_i$ . Since the communication complexity of the protocol is  $\mathcal{O}(b)$ , the sum of all  $\alpha_i$ 's should be bounded by  $db$ , for some constant  $d \geq 1$ . Hence

$$\begin{aligned} db \geq \sum_{i=1}^k \alpha_i &\geq \frac{\ell \log(t_f)}{\mathcal{D} \log(t_f)} \frac{X + t_f}{X + L_1} + \frac{\ell \log(t_f)}{\mathcal{D} \log(L_1)} \frac{X + L_1}{X + L_2} \\ &\quad + \dots + \frac{\ell \log(t_f)}{\mathcal{D} \log(L_{k-1})} \frac{X + L_{k-1}}{X + t_f^\epsilon} \end{aligned}$$

Since each  $\frac{1}{\log(L_i)} \geq \frac{1}{\log(t_f)}$ , we get

$$\begin{aligned}
db &\geq \frac{\ell \log(t_f)}{\mathcal{D} \log(t_f)} \left[ \frac{X+t_f}{X+L_1} + \frac{X+L_1}{X+L_2} + \dots + \frac{X+L_{k-1}}{X+t_f^\epsilon} \right]. \text{ Hence,} \\
\frac{db\mathcal{D}}{\ell} &\geq \frac{X+t_f}{X+L_1} + \frac{X+L_1}{X+L_2} + \dots + \frac{X+L_{k-1}}{X+t_f^\epsilon} \text{ and} \\
\frac{db\mathcal{D}}{\ell k} &\geq \frac{\left( \frac{X+t_f}{X+L_1} + \frac{X+L_1}{X+L_2} + \dots + \frac{X+L_{k-1}}{X+t_f^\epsilon} \right)}{k} \\
&\geq \left( \frac{X+t_f}{X+L_1} \times \frac{X+L_1}{X+L_2} \times \dots \times \frac{X+L_{k-1}}{X+t_f^\epsilon} \right)^{\frac{1}{k}}
\end{aligned}$$

The last inequality follows from the fact that Arithmetic Mean (AM)  $\geq$  Geometric Mean (GM). Since  $t_f > X$ , we get

$$\begin{aligned}
\left[ \frac{db\mathcal{D}}{\ell k} \right]^k &\geq \left[ \frac{X+t_f}{X+t_f^\epsilon} \right]^k \geq \frac{1}{2^k} \left( \frac{t_f}{X} \right)^{(1-\epsilon)k}, \\
\text{or equivalently, } \left[ \frac{2db\mathcal{D}}{\ell k} \right]^k &\geq \left( \frac{t_f}{X} \right)^{1-\epsilon}
\end{aligned}$$

Let  $a = 2d$ , where  $a \geq 2$ . Then we get

$$\left[ \frac{a b \log\left(\frac{t_f}{n-t_f}\right)}{\ell k \log\left(\frac{cb}{\ell}\right)} \right]^k - \left( \frac{t_f}{n-t_f} \right)^{1-\epsilon} \geq 0 \quad (5.1)$$

$$\text{Let } Y = \left[ \frac{a b \log\left(\frac{t_f}{n-t_f}\right)}{\ell k \log\left(\frac{cb}{\ell}\right)} \right]^k - \left( \frac{t_f}{n-t_f} \right)^{1-\epsilon}.$$

For our desired PRMT protocol, the value of  $k$  should be such that  $Y$  is non-negative. For this, we first find the range of  $k$  in which  $Y$  is an increasing function of  $k$ . Towards this, we make the following claim.

**Claim 5.15**  $Y = \left[ \frac{a b \log\left(\frac{t_f}{n-t_f}\right)}{\ell k \log\left(\frac{cb}{\ell}\right)} \right]^k - \left( \frac{t_f}{n-t_f} \right)^{1-\epsilon}$  is an increasing function for all  $k \leq$

$$\frac{ab \log\left(\frac{t_f}{n-t_f}\right)}{\ell \epsilon \log\left(\frac{cb}{\ell}\right)} = \frac{ab\mathcal{D}}{e \ell}$$

PROOF:

$$Y = \left[ \frac{a b \log\left(\frac{t_f}{n-t_f}\right)}{\ell k \log\left(\frac{cb}{\ell}\right)} \right]^k - \left( \frac{t_f}{n-t_f} \right)^{1-\epsilon} \quad (5.2)$$

$$Y = \left( \frac{Z}{k} \right)^k - Z' \quad \text{where } Z = \frac{a b \log\left(\frac{t_f}{n-t_f}\right)}{\ell \log\left(\frac{cb}{\ell}\right)} \text{ and } Z' = \left( \frac{t_f}{n-t_f} \right)^{1-\epsilon}$$

$$\Rightarrow \log(Y + Z') = k \log(Z) - k \log(k)$$

$$\Rightarrow \frac{1}{Y + Z'} \frac{dY}{dk} = \log(Z) - \log(k) - 1$$

$$\Rightarrow \frac{dY}{dk} = \left( \frac{Z}{k} \right)^k \left[ \log\left(\frac{Z}{k e}\right) \right]$$

Now putting  $\frac{dY}{dk} \geq 0$ , we get

$$\begin{aligned} \left(\frac{Z}{k}\right)^k \left[\log\left(\frac{Z}{k e}\right)\right] &\geq 0 \Rightarrow \frac{Z}{k e} \geq 1 \text{ as } \log 1 = 0 \\ \Rightarrow k &\leq \frac{Z}{e} = \frac{a b \log\left(\frac{t_f}{n-t_f}\right)}{e \ell \log\left(\frac{cb}{\ell}\right)} = \frac{ab\mathcal{D}}{e \ell} \square \end{aligned} \quad (5.3)$$

Thus we have shown that:

$$\forall k, k \leq \frac{ab\mathcal{D}}{e \ell}, Y \text{ is an increasing function.} \quad (5.4)$$

$$\text{Next we show that } Y \text{ is negative at } k = \frac{\mathcal{D}}{a c}. \quad (5.5)$$

Before that we prove the following:

**Claim 5.16**  $\frac{\mathcal{D}}{ac} < \frac{ab\mathcal{D}}{e \ell}$

PROOF: Recall that  $c \geq 1$  and  $a \geq 2$ . Also  $e = 2.73\dots$  Thus,

$$\begin{aligned} a &\geq 2 \\ \Rightarrow a^2 &\geq 4 \\ \Rightarrow 1 &\geq \frac{4}{a^2} \\ \Rightarrow c &> \frac{4}{a^2} \\ \Rightarrow c &> \frac{e}{a^2} \\ \Rightarrow \frac{a}{e} &> \frac{1}{ac} \end{aligned}$$

Hence  $\frac{\mathcal{D}}{ac} < \frac{a\mathcal{D}}{e} < \frac{ab\mathcal{D}}{e \ell}$ , since  $b \geq \ell$  □

**Lemma 5.17** *The value of  $Y = \left[\frac{ab\mathcal{D}}{\ell k}\right]^k - \left(\frac{t_f}{n-t_f}\right)^{1-\epsilon}$  is non-positive at  $k = \frac{\mathcal{D}}{ac}$  for all  $a \geq 2$  and some specific positive fraction  $\epsilon$ .*

PROOF: The proof is by contradiction. Assume  $Y$  to be non-negative at  $k = \frac{\mathcal{D}}{ac}$ . That is, at  $k = \frac{\mathcal{D}}{ac}$ , the inequality  $\left[\frac{ab\mathcal{D}}{\ell k}\right]^k \geq \left(\frac{t_f}{n-t_f}\right)^{1-\epsilon}$  holds. So, putting  $k = \frac{\mathcal{D}}{ac}$  in this relation and simplifying we get,

$$\begin{aligned} \left[\frac{a^2bc}{\ell}\right]^{\frac{\mathcal{D}}{ac}} &\geq \left(\frac{t_f}{n-t_f}\right)^{1-\epsilon} \\ \Rightarrow \frac{\mathcal{D}}{ac} \left[\log\left(\frac{cb}{\ell}\right) + 2\log(a)\right] &\geq (1-\epsilon) \log\left(\frac{t_f}{n-t_f}\right) \\ \Rightarrow \log\left(\frac{cb}{\ell}\right) \left[\frac{1}{ac} - (1-\epsilon)\right] + \frac{2\log(a)}{ac} &\geq 0 \text{ as } \log\left(\frac{t_f}{n-t_f}\right) = \mathcal{D} \log\left(\frac{cb}{\ell}\right) \end{aligned}$$

Now this relation does not hold for all values of  $a \geq 2$  for a specified choice of  $\epsilon$ . Specifically, for  $c = a^{10} + 1$  and  $\epsilon = 0.4$ , the equation does not hold. Hence this is a contradiction that  $Y$  is non-negative at  $k = \frac{\mathcal{D}}{ac}$ . Hence the lemma. □

Now equation (5.4) and (5.5) together imply that for all values of  $k < \frac{\mathcal{D}}{ac}$ , the value of

$Y$  is negative. Hence, for the value of  $Y$  to be non-negative,  $k$  should satisfy  $k > \frac{\mathcal{D}}{ac} = \Omega(\mathcal{D})$ . This establishes the lower bound on the phase complexity.

Thus, in all three cases, we proved that the number of phases required is  $\Omega(\mathcal{D})$ . This completes the proof of Theorem 5.10.  $\square$

We now dispose two important corollaries.

**Corollary 5.17.1** *Any PRMT protocol over  $n = t_f + 1$  wires, influenced by  $\mathcal{A}_{t_f}^{static}$  (i.e.,  $t_b = 0$ ), must run for  $\Omega(\log(t_f))$  phases to reliably send  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements.*

Thus against only fail-stop adversary over a minimal connected network, it takes  $\log(t_f)$  phases to achieve reliability with *constant factor* overhead. Theorem 3.16 and Corollary 5.17.1 define the phase complexity lower bounds for two extreme cases (namely when  $t_f = 0$  and  $t_b = 0$  respectively). The intermediate scenario is captured by the following corollary which brings out the fundamental inherent trade-off between phase complexity and communication complexity in the presence of  $\mathcal{A}_{(t_b, t_f)}^{static}$ .

**Corollary 5.17.2** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $2t_b + t_f + 1$  wires, such that  $t_b, t_f > 0$  and  $t_f > (n - t_f)$ . Then any PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$  must run for  $\Omega(\log(\frac{t_f}{t_b}))$  phases to reliably send  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements.*

## 5.8 Upper Bound on Phase Complexity of PRMT Tolerating $\mathcal{A}_{(t_b, t_f)}^{static}$

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + t_f + 1$  wires. We then provide a PRMT protocol called **Optimal-Static-PRMT-Mixed**, which terminates in  $\mathcal{O}(\mathcal{D})$  phases, tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ . The protocol sends a *sufficiently large* message  $m^{\mathbf{S}}$  containing  $\ell$  field elements ( $\ell$  will be fixed shortly) by communicating  $\mathcal{O}(b)$  field elements where  $\ell \leq b < n\ell$  and  $\mathcal{D} = \frac{\log(\frac{t_f}{n-t_f})}{\log(\frac{cb}{\ell})}$ . This shows that the bound proved in Theorem 5.10 is asymptotically tight. Before presenting the protocol, we first design a three phase PRMT protocol called **3-PRMT-Static-Mixed**, which is used in protocol **Optimal-Static-PRMT-Mixed**.

### 5.8.1 Protocol 3-PRMT-Static-Mixed: A Three Phase PRMT Protocol

In protocol **3-PRMT-Static-Mixed**,  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $N \geq 2t_b + T + 1$  wires, of which at most  $t_b$  could be Byzantine corrupted and  $T \leq t_f$  could be fail-stop corrupted. Also  $N$  and  $T$  are such that  $(n - t_f) = (N - T)$ . Let  $(n - t_f) = (N - T) = X$ . The protocol reliably sends a message  $M$  containing  $L$  field elements by communicating  $\mathcal{O}(\frac{NL}{X})$  field elements, where  $L \geq N^2$ . Note that in **3-PRMT-Static-Mixed**, we have used  $M, L, N$  and  $T$  instead of  $m^{\mathbf{S}}, \ell, n$  and  $t_f$  respectively. This is because **3-PRMT-Static-Mixed** will be used as a black-box in protocol **Optimal-Static-PRMT-Mixed** several times. Moreover each time **3-PRMT-Static-Mixed** is used in **Optimal-Static-PRMT-Mixed**, the value of  $M, L, N$  and  $T$  may vary.

Protocol **3-PRMT-Static-Mixed** uses the error correction and detection capability of RS codes. The high level idea of the protocol is as follows:  $\mathbf{S}$  divides  $M$  into several blocks of size  $(X - 2t_b) + \frac{t_b}{2}$  and tries to reliably send  $M$  by encoding each block of  $M$  into an RS codeword of length  $N$ . If at most  $\frac{t_b}{2}$  Byzantine errors occur during the transmission, then  $\mathbf{R}$  will successfully recover  $M$ . Otherwise,  $\mathbf{R}$  will detect that more than  $\frac{t_b}{2}$  Byzantine errors have occurred. In this case,  $\mathbf{R}$  sends back the received

information to  $\mathbf{S}$ , who then finds the identity of those corrupted wires (which are more than  $\frac{t_b}{2}$ ).  $\mathbf{S}$  then reliably sends the identity of those corrupted wires to  $\mathbf{R}$ .  $\mathbf{S}$  then again re-sends  $M$  by using protocol 1-PRMT-Mixed with increased throughput. The protocol is formally presented in Fig. 5.3.

Figure 5.3: A Three Phase PRMT Protocol Tolerating Mixed Adversary

**3-PRMT-Static-Mixed**( $M, L, N, t_b, T$ )

**Phase I: S to R:**

1. Let  $X = (N - T)$ .  $\mathbf{S}$  divides  $M$  into blocks  $\mathbf{B}_1^{\mathbf{S}}, \dots, \mathbf{B}_z^{\mathbf{S}}$ , each of size  $(X - 2t_b + \frac{t_b}{2})$ .
2. For  $j = 1, \dots, z$ , corresponding to block  $\mathbf{B}_j^{\mathbf{S}}$ ,  $\mathbf{S}$  computes an RS codeword of size  $N$ , denoted by  $C_j^{\mathbf{S}} = (c_{j1}^{\mathbf{S}}, \dots, c_{jN}^{\mathbf{S}})$ .
3. For  $i = 1, \dots, N$ , sends  $c_{ji}^{\mathbf{S}}$  to  $\mathbf{R}$  through wire  $w_i$ , for  $j = 1, \dots, z$ .

**Phase II: R to S**

1. Let  $\mathbf{R}$  receive information over wires  $w_{i_1}, \dots, w_{i_{N'}}$ , where  $\{w_{i_1}, \dots, w_{i_{N'}}\} \subseteq \{w_1, \dots, w_N\}$  and  $N' \geq N - T$ . For  $j = 1, \dots, z$ , let  $\mathbf{R}$  receive  $c_{ji_1}^{\mathbf{R}}, \dots, c_{ji_{N'}}^{\mathbf{R}}$  along wire  $w_{i_1}, \dots, w_{i_{N'}}$  respectively. Let  $C_j^{\mathbf{R}} = (c_{ji_1}^{\mathbf{R}}, \dots, c_{ji_{N'}}^{\mathbf{R}})$ .
2. For  $j = 1, \dots, z$ ,  $\mathbf{R}$  executes RS-DEC( $N', C_j^{\mathbf{R}}, \frac{t_b}{2}, \frac{t_b}{2}, (X - 2t_b + \frac{t_b}{2})$ ).
3. If after correcting  $\frac{t_b}{2}$  errors, the decoding algorithm does not detect additional faults in any of the  $z$  received vectors, then  $\mathbf{R}$  correctly recovers  $\mathbf{B}_j^{\mathbf{S}}$ ,  $1 \leq j \leq z$ .  $\mathbf{R}$  then concatenates them to recover  $M$ , broadcasts "SUCCESS" signal to  $\mathbf{S}$  and terminates the protocol. This case implies that at most  $\frac{t_b}{2}$  Byzantine errors have occurred during transmission by  $\mathbf{S}$ .
4. If  $\exists e \in \{1, 2, \dots, z\}$ , such that after correcting  $\frac{t_b}{2}$  errors, the decoding algorithm detects additional faults (at most  $\frac{t_b}{2}$ ) in the  $e^{th}$  received vector  $C_e^{\mathbf{R}}$ , then  $\mathbf{R}$  broadcasts  $C_e^{\mathbf{R}}$ , along with index  $e$  and "ERROR" signal to  $\mathbf{S}$ . If there are several such  $e$ 's, then  $\mathbf{R}$  randomly selects one.  $\mathbf{R}$  also broadcasts the identity of wires which failed to deliver any information. This case implies that more than  $\frac{t_b}{2}$  Byzantine errors have occurred during transmission by  $\mathbf{S}$ .

If  $\mathbf{S}$  receives "SUCCESS" signal, then  $\mathbf{S}$  terminates the protocol. ELSE  $\mathbf{S}$  receives "ERROR" signal, index  $e$ , vector  $C_e^{\mathbf{R}}$  as received by  $\mathbf{R}$  in **Phase I** and initiates **Phase III** as follows:

**Phase III: S to R:**

1. After comparing  $C_e^{\mathbf{S}}$  (sent during **Phase I**) with  $C_e^{\mathbf{R}}$  (received by  $\mathbf{R}$  at the end of **Phase I**),  $\mathbf{S}$  identifies more than  $\frac{t_b}{2}$  faulty wires (which delivered incorrect information to  $\mathbf{R}$  during **Phase I**).
2.  $\mathbf{S}$  saves the identity of corrupted wires in a list  $L_{fault}$ .  $\mathbf{S}$  then broadcasts  $L_{fault}$  to  $\mathbf{R}$ .
3.  $\mathbf{S}$  re-sends  $M$  by executing 1-PRMT-Mixed( $M, L, N, t_b, T, |L_{fault}|$ ) with increased throughput and terminates the protocol.

**Local Computation by R**

1.  $\mathbf{R}$  correctly receives  $L_{fault}$  and identifies  $|L_{fault}| > \frac{t_b}{2}$  Byzantine corrupted wires.
2.  $\mathbf{R}$  finally recovers  $M$  at the end of 1-PRMT-Mixed( $M, L, N, t_b, T, |L_{fault}|$ ) and terminates the protocol.

We now prove the properties of protocol 3-PRMT-Static-Mixed.

**Theorem 5.18 (Correctness)** *In protocol 3-PRMT-Static-Mixed,  $\mathbf{R}$  will correctly recover  $M$  in at most three phases.*

PROOF: We prove the theorem for the worst case, when during **Phase I**, the fail-stop adversary blocks all  $T$  wires under its control. So during **Phase I**,  $\mathbf{R}$  receives  $N' = N - T$  values for each  $\mathbf{B}_j^{\mathbf{S}}, 1 \leq j \leq z$ , which are RS encoded using polynomials of

degree  $k - 1 = (X - 2t_b) + \frac{t_b}{2} - 1$ . Putting the values of  $N'$  and  $k$  in Theorem 2.19, we find that RS decoding can correct  $c = \frac{t_b}{2}$  errors and simultaneously detect additional  $d = \frac{t_b}{2}$  errors in each of the  $z$  received vectors. Now there are following two cases:

1. *At most  $\frac{t_b}{2}$  Byzantine errors occur during **Phase I**:* In this case, the decoding algorithm will correct these errors and will not detect additional errors in any received vector. So  $\mathbf{R}$  will recover each  $\mathbf{B}_j^{\mathbf{S}}$  (and hence  $M$ ) at the end of **Phase I**. Moreover,  $\mathbf{S}$  will also come to know this at the end of **Phase II** after receiving the "SUCCESS" signal.
2. *More than  $\frac{t_b}{2}$  Byzantine errors occur during **Phase I**:* In this case,  $\exists e \in \{1, 2, \dots, z\}$ , such that  $e^{\text{th}}$  received vector  $C_e^{\mathbf{R}}$  contains more than  $\frac{t_b}{2}$  corrupted values. So, RS decoding will correct  $\frac{t_b}{2}$  errors and simultaneously detect the remaining faults (which are at most  $\frac{t_b}{2}$ ). So  $\mathbf{R}$  will come to know that more than  $\frac{t_b}{2}$  values are corrupted in  $C_e^{\mathbf{R}}$ . In this case,  $\mathbf{R}$  reliably sends back  $C_e^{\mathbf{R}}$  to  $\mathbf{S}$ , who after comparing it with the original codeword  $C_e^{\mathbf{S}}$ , finds the identity of more than  $\frac{t_b}{2}$  Byzantine faults and saves them in  $L_{\text{fault}}$ . By broadcasting  $L_{\text{fault}}$ ,  $\mathbf{S}$  informs the identity of these faulty wires to  $\mathbf{R}$ . Finally  $\mathbf{S}$  re-sends  $M$  by executing 1-PRMT-Mixed( $M, L, N, t_b, T, |L_{\text{fault}}|$ ) (with increased throughput). From  $L_{\text{fault}}$ ,  $\mathbf{R}$  will come to know the identity of  $|L_{\text{fault}}|$  corrupted wires. So from Theorem 5.7, 1-PRMT-Mixed will be executed successfully and hence  $\mathbf{R}$  will recover  $M$  at the end of **Phase III**.  $\square$

**Theorem 5.19 (Communication Complexity)** *The communication complexity of protocol 3-PRMT-Static-Mixed is  $\mathcal{O}(\frac{NL}{X})$ , where  $X = (N - T)$  and  $L \geq N^2$ .*

PROOF: During **Phase I**,  $\mathbf{S}$  communicates  $\mathcal{O}\left(\frac{|M|}{(X-2t_b)+\frac{t_b}{2}} \times N\right) = \mathcal{O}(\frac{NL}{X})$  field elements. During **Phase II**,  $\mathbf{R}$  may communicate  $\mathcal{O}(N^2)$  field elements by broadcasting the  $e^{\text{th}}$  received vector. During **Phase III** (if it is executed),  $\mathbf{S}$  re-sends  $M$  by executing PRMT-Mixed( $M, L, N, t_b, T, |L_{\text{fault}}|$ ), which from Theorem 5.7 communicates  $\mathcal{O}\left(\frac{|M|}{(X-2t_b)+|L_{\text{fault}}|} \times N\right) = \mathcal{O}(\frac{NL}{X})$  field elements because  $\frac{t_b}{2} < |L_{\text{fault}}| \leq t_b$ . Since,  $L \geq N^2$ , the total communication complexity is  $\mathcal{O}(\frac{NL}{X})$ .  $\square$

We now give two important corollaries of Theorem 5.19.

- Corollary 5.19.1** *1. If  $X \geq T$ , then  $X = \Theta(N)$  because  $N = X + T$ <sup>2</sup>. Hence, in this case protocol 3-PRMT-Static-Mixed sends  $L$  field elements by communicating  $\mathcal{O}(L)$  field elements, where  $L \geq N^2$ .*
- 2. If  $T > X$ , then  $T = \Theta(N)$  because  $N = X + T$ . So in this case protocol 3-PRMT-Static-Mixed sends  $L$  field elements by communicating  $\mathcal{O}(\frac{LT}{X})$  field elements, where  $L \geq N^2$ .*

### 5.8.2 A Worst Case $\mathcal{O}(\mathcal{D})$ Phase PRMT Protocol Tolerating $\mathcal{A}_{(t_b, t_f)}^{\text{static}}$

Let  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n \geq 2t_b + t_f + 1$  wires. Then we present a protocol called Optimal-Static-PRMT-Mixed tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{static}}$ , which has the following properties:

<sup>2</sup>From [21], if  $f(n)$  and  $g(n)$  are asymptotically non-negative functions then  $\max(f(n), g(n)) = \Theta(f(n) + g(n))$ .

1. If  $(n - t_f) \geq t_f$  then the protocol sends  $m^{\mathbf{S}}$  in three phases by communicating  $\mathcal{O}(|m^{\mathbf{S}}|)$  field elements.
2. If  $(n - t_f) < t_f$ , then depending upon the adversary behavior, there are two possibilities:
  - (a) The protocol reliably sends  $m^{\mathbf{S}}$  in  $\mathcal{O}(\mathcal{D})$  phases by communicating  $\mathcal{O}(b)$  field elements, where  $|m^{\mathbf{S}}| = \ell$ ,  $\mathcal{D} = \frac{\log(\frac{t_f}{n-t_f})}{\log(\frac{cb}{\ell})}$  and  $\ell \leq b < n\ell$ .
  - (b) The protocol fails to send  $m^{\mathbf{S}}$ . But it reduces  $n$  and  $t_f$  such that  $(n-t_f) \geq t_f$ . This process will take  $\mathcal{O}(\mathcal{D})$  phases and requires a communication complexity of  $\mathcal{O}(b)$  field elements, where  $\ell \leq b < n\ell$ . Once  $n$  and  $t_f$  satisfies  $(n-t_f) \geq t_f$ ,  $m^{\mathbf{S}}$  is sent using the three phase protocol 3-PRMT-Static-Mixed.

To design the protocol, we use protocol 2-SP-REL-Mixed (given in Section 5.6) as a black-box. In **Optimal-Static-PRMT-Mixed**,  $N$  denote the number of wires between  $\mathbf{S}$  and  $\mathbf{R}$  and  $T$  denote the number of undisclosed fail-stop corruption. Both  $N$  and  $T$  are global variables. Initially,  $N = n$  and  $T = t_f$ .  $\mathbf{S}$  and  $\mathbf{R}$  also maintains a list of current wires between  $\mathbf{S}$  and  $\mathbf{R}$ . Each time some wire is identified to be fail-stop corrupted, it is removed from the list of current wires and accordingly the value of  $N$  and  $T$  are reduced. The protocol is formally given in Fig. 5.4.

**Remark 5.20 (Size of  $m^{\mathbf{S}}$  in **Optimal-Static-PRMT-Mixed**)** *If **Optimal-Static-PRMT-Mixed** calls **3-PRMT-Static-Mixed** to send  $M'$ , then  $M'$  will contain at least one chunk of  $m^{\mathbf{S}}$  of size  $\frac{\ell}{\mathcal{D}}$ . Since **3-PRMT-Static-Mixed** requires the minimum message size ( $L$ ) to be  $N^2$  (where  $N \leq n$ ), we take  $|m| = \ell = n^3$ , which ensures that  $|M'| = \frac{n^3}{\mathcal{D}} \geq n^2 \geq N^2$ .*

The analysis of **Optimal-Static-PRMT-Mixed** is divided into following two cases:

1. When  $X \geq T$ ;
2. When  $X < T$ . This case has further two sub-cases, depending upon whether  $\mathcal{D} \geq 1$  or  $\mathcal{D} < 1$ .

We now consider all the above cases one by one.

**Lemma 5.21** *If  $X \geq T$ , then **Optimal-Static-PRMT-Mixed** reliably sends  $m^{\mathbf{S}}$  in three phases by communicating  $\mathcal{O}(\ell)$  field elements.*

PROOF: Follows from Part(1) of Corollary 5.19.1 by substituting  $L = |m^{\mathbf{S}}|$ , where  $|m^{\mathbf{S}}| = n^3$ .  $\square$

**Lemma 5.22** *If  $X < T$  and  $\mathcal{D} < 1$  then **Optimal-Static-PRMT-Mixed** reliably sends  $m^{\mathbf{S}}$  in three phases by communicating  $\mathcal{O}(b)$  field elements.*

PROOF: If  $X < T$  then  $N = \Theta(T)$  because  $N = X + T$ . Also if  $\mathcal{D} < 1$  then it implies  $\log(\frac{cb}{\ell}) > \log(\frac{t_f}{n-t_f}) \Rightarrow \frac{cb}{\ell} > \frac{t_f}{n-t_f} \Rightarrow b > \frac{t_f \ell}{c(n-t_f)} \Rightarrow b > \frac{T\ell}{cX}$ . This is because initially  $N = n$  and  $T = t_f$ . Also  $X = (N - T)$ . Thus, we require to send  $\ell$  field elements by communicating  $\mathcal{O}(\frac{T\ell}{cX})$  field elements. Since  $\ell = n^3 = N^3$ , this implies that **Optimal-Static-PRMT-Mixed** has to send  $N^3$  field elements by communicating  $\mathcal{O}(\frac{N^3 T}{cX})$  field elements. From Part(2) of Corollary 5.19.1, we find that this can be done in three

phases by executing 3-PRMT-Static-Mixed.  $\square$

Lemma 5.21 and Lemma 5.22 prove the properties of Optimal-Static-PRMT-Mixed for two cases. Now we analyze the properties of Optimal-Static-PRMT-Mixed for the case when  $X < T$  and  $\mathcal{D} \geq 1$ . In this case, the execution sequence of Optimal-Static-PRMT-Mixed is as follows:  $\mathbf{S}$  sequentially selects a chunk  $B_i$  of size  $\frac{\ell}{\mathcal{D}}$  from the message  $m^{\mathbf{S}}$ .  $\mathbf{S}$  then executes 2-SP-REL-Mixed to send  $B_i$  using  $(X - 2t_b) + \frac{t_b}{2} + k_f$  as the block-size, where  $k_f = T \frac{\ell}{cb}$ . In this process either (a)  $\mathbf{R}$  recovers  $B_i$  or (b)  $\mathbf{S}$  and  $\mathbf{R}$  identifies at least  $T - k_f + 1$  fail-stop corrupted wires or more than  $\frac{t_b}{2}$  Byzantine corrupted wires, which subsequently  $\mathbf{S}$  and  $\mathbf{R}$  removes from their list of current wires. Accordingly, the value of  $N, T$  and  $X$  are updated. In the first case  $\mathbf{R}$  receives  $B_i$  and then  $\mathbf{S}$  selects the next chunk  $B_{i+1}$  and repeats the same process. In the later cases, depending upon the type of identified faulty wires,  $\mathbf{S}$  re-sets the block-size and tries to re-send the same  $B_i$  by executing 2-SP-REL-Mixed. Note that if more than  $\frac{t_b}{2}$  Byzantine corrupted wires are identified, then the same block size is used to re-send  $B_i$ , whereas if more than  $T - k_f$  fail-stop corrupted wires are identified then block-size becomes  $(X - 2t_b) + \frac{t_b}{2} + k_f$ , where  $k_f$  is reduced at least by a factor of  $(\frac{cb}{\ell})$ . This process will continue until the entire  $m^{\mathbf{S}}$  is received by  $\mathbf{R}$  or the number of unknown fail-stop faults  $T$  becomes less than or equal to  $X$ . When number of unknown fail-stop faults  $T$  becomes less than or equal to  $X$ ,  $\mathbf{S}$  sends the remaining portion of  $m^{\mathbf{S}}$ , say  $\hat{m}^{\mathbf{S}}$ , by executing the three phase PRMT protocol 3-PRMT-Static-Mixed, which will optimally send  $\hat{m}^{\mathbf{S}}$  by communicating  $O(|\hat{m}^{\mathbf{S}}|)$  field elements.

Thus, in summary, the execution sequence for this case is as follows: we create a win-win situation with the adversary, such that if the number of fail-stop corruptions done by the adversary is less than a specific limit then  $\mathbf{S}$  will be able to successfully send  $B_i$  to  $\mathbf{R}$ . Moreover, the size of  $B_i$  will be more than what  $\mathbf{S}$  is allowed to send in a single phase to  $\mathbf{R}$  according to Singleton Bound. On the other hand, if the the number of fail-stop corruptions done by the adversary is more than the specific limit then  $\mathbf{S}$  will be not be able to successfully send  $B_i$  to  $\mathbf{R}$ . But in this case, the number of unknown fail-stop corruption, i.e.  $T$ , is reduced at least by a factor of  $\frac{cb}{\ell}$ . This process is repeated till either the entire  $m^{\mathbf{S}}$  is delivered or the value of  $T$  (after several reductions) becomes less than or equal to that of  $X$ . In the later case,  $m^{\mathbf{S}}$  is re-sent using 3-PRMT-Static-Mixed. Interestingly, we will show (in the sequel) that this entire process takes  $\mathcal{O}(\mathcal{D})$  phases and a communication complexity of  $\mathcal{O}(b)$ .

**Lemma 5.23** *In protocol Optimal-Static-PRMT-Mixed, if some execution of 2-SP-REL-Mixed fails due to the occurrence of more than  $\frac{t_b}{2}$  Byzantine errors, then the remaining executions of 2-SP-REL-Mixed in Optimal-Static-PRMT-Mixed can fail only due to occurrence of more than  $T - k_f$  fail-stop errors.*

PROOF: In protocol Optimal-Static-PRMT-Mixed if some execution of 2-SP-REL-Mixed fails due to the occurrence of more than  $\frac{t_b}{2}$  Byzantine errors, then both  $\mathbf{S}$  and  $\mathbf{R}$  will know the identity of these wires. This implies that in the remaining executions of 2-SP-REL-Mixed in Optimal-Static-PRMT-Mixed, at most  $\frac{t_b}{2} - 1$  Byzantine errors can occur. If in all these remaining executions of 2-SP-REL-Mixed, at most  $T - k_f$  fail-stop errors occur, then in each execution of 2-SP-REL-Mixed,  $\mathbf{R}$  will receive  $N' = N - (T - k_f)$  values for each sub-block of  $B_i$ , where each sub-block is RS encoded using a polynomial of degree  $k - 1 = (X - 2t_b) + \frac{t_b}{2} + k_f - 1$ . Putting these values in Theorem 2.19, we find that RS decoding algorithm will be able to correct  $c = \frac{t_b}{2}$  errors in the received vectors and hence 2-SP-REL-Mixed will be successful. So if some execution of 2-SP-REL-Mixed fails then it will be due to the occurrence of more than  $T - k_f$  fail-stop errors.  $\square$

**Lemma 5.24** *In protocol **Optimal-Static-PRMT-Mixed**, if any execution of **2-SP-REL-Mixed** fails due to occurrence of more than  $T - k_f$  fail-stop errors, then  $T$  is reduced at least by a factor of  $\frac{cb}{\ell}$ .*

PROOF: From Lemma 5.8, if **2-SP-REL-Mixed** fails due to the occurrence of more than  $T - k_f$  fail-stop errors, then the number of unknown fail-stop errors reduces to at least  $T - (T - k_f + 1) = k_f - 1 = T \frac{\ell}{cb} - 1$ .  $\square$

**Theorem 5.25** *If  $X < T$  and  $\mathcal{D} \geq 1$  then **Optimal-Static-PRMT-Mixed** terminates in  $\mathcal{O}(\mathcal{D})$  phases.*

PROOF: If  $X < T$  and  $\mathcal{D} \geq 1$  then the phase complexity of **Optimal-Static-PRMT-Mixed** is bounded by the number of times protocol **2-SP-REL-Mixed** is executed in **Optimal-Static-PRMT-Mixed**. There are  $q = \mathcal{D}$  chunks of  $m^{\mathbf{S}}$  and **2-SP-REL-Mixed** is executed at least once to send each of them. If each chunk is sent successfully in a single attempt, then the phase complexity of **Optimal-Static-PRMT-Mixed** is  $\mathcal{O}(\mathcal{D})$ .

On the other hand, from Lemma 5.24, each un-successful execution of **2-SP-REL-Mixed** in protocol **Optimal-Static-PRMT-Mixed** due to occurrence of more than  $T - k_f$  fail-stop errors reduces the number of unknown fail-stop errors  $T$  at least by a factor of  $\frac{cb}{\ell}$ . Also, from Lemma 5.23, **2-SP-REL-Mixed** can fail only *once* due to Byzantine errors. Hence the number of un-successful executions of **2-SP-REL-Mixed** required to bring the number of unknown fail-stop errors  $T$ , from its initial value of  $t_f$  to  $n - t_f (= X)$  is

bounded by  $\mathcal{O}\left(\frac{\log\left(\frac{t_f}{n-t_f}\right)}{\log\left(\frac{cb}{\ell}\right)}\right) = \mathcal{O}(\mathcal{D})$ . Once  $T$  becomes less than or equal to  $X$ , the

remaining message is sent in three phases by executing **3-PRMT-Static-Mixed**. Thus even if adversary alternately allows some (un)successful executions of **2-SP-REL-Mixed** followed by some unsuccessful(successful) executions of **2-SP-REL-Mixed**, the phase complexity of **Optimal-Static-PRMT-Mixed** is bounded by  $\mathcal{O}(\mathcal{D})$ .  $\square$

**Theorem 5.26** *The communication complexity of **Optimal-Static-PRMT-Mixed** is  $\mathcal{O}(b)$ , where  $\ell \leq b < n\ell$  and  $\ell \geq n^3$ .*

PROOF: In **Optimal-Static-PRMT-Mixed**, if step 2 is executed, then from Lemma 5.21, the communication complexity of the protocol is  $\mathcal{O}(\ell) = \mathcal{O}(b)$ . Similarly, if step 3 is executed, then from Lemma 5.22, the communication complexity of the protocol is  $\mathcal{O}(b)$ . However, if step 4 is executed, then the communication complexity of **Optimal-Static-PRMT-Mixed** is computed as follows:

To send a chunk  $B_i$  of size  $\frac{\ell}{\mathcal{D}}$ ,  $\mathbf{S}$  executes **2-SP-REL-Mixed** with block-size  $k = (X - 2t_b) + \frac{t_b}{2} + k_f = (X - 2t_b) + \frac{t_b}{2} + T \frac{\ell}{cb}$ . Also at every stage of the protocol  $N = T + X$ . From Lemma 5.9, the number of field elements sent by  $\mathbf{S}$  during the execution of **2-SP-REL-Mixed** is given by  $\frac{\frac{\ell}{\mathcal{D}}}{(X - 2t_b) + \frac{t_b}{2} + T \frac{\ell}{cb}}(X + T)$ . Since,  $T > X$ , by increasing the numerator and decreasing the denominator, the above expression is bounded by  $\frac{\ell}{\mathcal{D}} \frac{2T}{T \frac{\ell}{cb}} = \frac{2cb}{\mathcal{D}} = \mathcal{O}\left(\frac{b}{\mathcal{D}}\right)$ . From Theorem 5.25, the overall phase complexity of **Optimal-Static-PRMT-Mixed** is  $\mathcal{O}(\mathcal{D})$ . Thus, the total number of field elements communicated by  $\mathbf{S}$  is  $\mathcal{O}(b)$ . In the protocol, each time **2-SP-REL-Mixed** fails,  $\mathbf{R}$  broadcasts a bit vector of length  $n$ , which requires communicating  $\mathcal{O}(n^2)$  field elements. Since, in **Optimal-Static-PRMT-Mixed**, the number of failures of **2-SP-REL-Mixed** is bounded by  $\mathcal{O}(\mathcal{D})$ , the total number of field elements communicated by  $\mathbf{R}$  in the protocol is  $\mathcal{O}(n^2\mathcal{D})$ . Since  $|m^{\mathbf{S}}| = n^3$  and  $b \geq |m^{\mathbf{S}}|$ , the overall communication complexity of **Optimal-Static-PRMT-Mixed** is  $\mathcal{O}(b) + \mathcal{O}(n^2\mathcal{D}) = \mathcal{O}(b)$ .  $\square$

**Theorem 5.27** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + t_f + 1$  wires, such that  $t_f > 0$  and  $t_f > (n - t_f)$ . Then there exists an  $\mathcal{O}(\mathcal{D})$  phase PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ , which sends a message  $m^{\mathbf{S}}$  containing  $\ell = \Theta(n^3)$  field elements by communicating  $\mathcal{O}(b)$  field elements, where  $\ell \leq b < n\ell$  and  $\mathcal{D} = \left( \frac{\log(\frac{t_f}{n-t_f})}{\log(\frac{cb}{\ell})} \right)$ .*

PROOF: Follows from the above discussion.  $\square$

## 5.9 Concluding Remarks and Open Problems

In this chapter we have studied the inherent tradeoff among all the important parameters of any PRMT protocol, namely network connectivity ( $n$ ), phase complexity ( $r$ ) and communication complexity ( $b$ ), in the presence of  $\mathcal{A}_{(t_b, t_f)}^{static}$ . Specifically, we have solved the holy grail problem of PRMT by deriving a non-trivial lower bound on the phase complexity of any PRMT protocol, which sends a given message over a given network in the presence of  $\mathcal{A}_{(t_b, t_f)}^{static}$ , such that the communication complexity of the protocol is bounded by a given limit. Our lower bound is first of its kind and shows the inherent tradeoff among all the parameters of any PRMT protocol simultaneously. Moreover, we have shown that our bound is asymptotically tight by presenting a PRMT protocol. As a bi-product of our proposed PRMT protocol, we get a PRMT protocol which is simultaneously optimal in all the parameters, namely  $n, b$  and  $r$ .

Our proposed PRMT protocol is optimal only for messages of some specific length (specifically if  $|m| = \Theta(n^3)$ ). This leads to the following open question:

**Open Problem 4** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + t_f + 1$  wires, such that  $t_f > 0$  and  $t_f > (n - t_f)$ . Let  $m^{\mathbf{S}}$  be a message containing  $\ell \geq 1$  field element(s) and let  $\ell \leq b < n\ell$ . Then does there exist an  $\mathcal{O}(\mathcal{D})$  phase PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ , which sends  $m^{\mathbf{S}}$  by communicating  $\mathcal{O}(b)$  field elements, for any value of  $\ell$ ?*

Figure 5.4: An  $\mathcal{O}(\mathcal{D})$  Phase PRMT Protocol Tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$

**Protocol Optimal-Static-PRMT-Mixed** ( $m^{\mathbf{S}}, n, t_b, t_f$ )

1. **S** and **R** initializes  $N = n$  and  $T = t_f$ . Let  $X = N - T$ .
2. IF  $X \geq T$ , then **S** sends  $m^{\mathbf{S}}$  by executing the three phase protocol 3-PRMT-Static-Mixed and terminate Optimal-Static-PRMT-Mixed (see Lemma 5.21).
3. IF ( $X < T$ ) AND ( $\mathcal{D} < 1$ ), then **S** sends  $m^{\mathbf{S}}$  by executing the three phase protocol 3-PRMT-Static-Mixed and terminate Optimal-Static-PRMT-Mixed (see Lemma 5.22).
4. IF ( $X < T$ ) AND ( $\mathcal{D} \geq 1$ ), then **S** and **R** does the following:
  - (a) **S** and **R** initializes  $i = 1$ . **S** breaks  $m^{\mathbf{S}}$  into chunks  $B_1, B_2, \dots, B_q$  each of size  $\frac{\ell}{\mathcal{D}}$  (so  $q = \mathcal{D}$ ).
  - (b) While ( $i \leq q$ ) AND ( $T > X$ ) DO
    - i. **S** sets  $k_f = T \frac{\ell}{c_b}$  and executes 2-SP-REL-Mixed( $B_i, N, t_b, T, k_f$ ) by using the block size as  $k = (X - 2t_b) + \frac{t_b}{2} + k_f$  and waits for feed-back. /\* Recall that  $k$  is the block-size which is used in 2-SP-REL-Mixed.\*/
    - ii. Depending upon the feedback that **S** receives at the end of 2-SP-REL-Mixed( $B_i, N, t_b, T, k_f$ ), **S** does the following:
      - A. **S** receives "SUCCESS" SIGNAL: In this case, **S** concludes that **R** has correctly received  $B_i$ . So both **S** and **R** increments  $i$  and continues with the next iteration.
      - B. **S** receives "ERROR1" SIGNAL: In this case, **S** concludes that **R** has failed to recover  $B_i$  because more than  $T - k_f$  fail-stop corruptions have occurred. In this case, **S** and **R** will also come to know the identity of  $N - N'$  fail-stop corrupted wires (here  $N'$  is the number of wires over which **R** has received information during 2-SP-REL-Mixed( $B_i, N, t_b, T, k_f$ )). **S** and **R** remove these wires from the list of current wires between **S** and **R** and does not consider them for the rest of the protocol. Moreover, **S** and **R** also reduces  $N$  and  $T$  and continues with the next iteration.
      - C. **S** receives "ERROR2" SIGNAL: In this case, **S** concludes that **R** has failed to recover  $B_i$  because more than  $\frac{t_b}{2}$  Byzantine corruptions have occurred during the execution of 2-SP-REL-Mixed( $B_i, N, t_b, T, k_f$ ). Moreover, **S** will also come to know their identity after doing local comparison. **S** saves their identity in a list  $L_{fault}$ . **S** then broadcasts  $L_{fault}$  to **R**, who also then identifies  $|L_{fault}| > \frac{t_b}{2}$  Byzantine corrupted wires. **S** and **R** then globally set  $N = N - |L_{fault}|$  and remove the wires in  $L_{fault}$  from the list of current wires between **S** and **R** and continue with the next iteration. Now number of Byzantine faulty wires in the set of current wires is less than  $\frac{t_b}{2}$ .
  - (c) If  $i > q$  then **S** and **R** terminates the protocol. Else **S** sends the remaining portion of the message, say  $M'$ , consisting of chunks  $B_i, B_{i+1}, \dots, B_q$  by executing the three phase PRMT protocol 3-PRMT-Static-Mixed.

## Chapter 6

# PRMT in Undirected Networks Tolerating Mobile Mixed Adversary

In this chapter, we discuss about PRMT in *undirected synchronous network, tolerating threshold mobile mixed adversary*. The mobile mixed adversary, denoted by  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ , may corrupt different set of  $t_b$  and  $t_f$  nodes in Byzantine and failstop fashion respectively, during different phases of the protocol. In Chapter 4, we have studied PRMT tolerating  $\mathcal{A}_{t_b}^{mobile}$  and showed that the mobility of the adversary has no effect on the POSSIBILITY and OPTIMALITY of PRMT protocols. That is, the necessary and sufficient condition are same for tolerating  $\mathcal{A}_{t_b}^{mobile}$ , as well as  $\mathcal{A}_{t_b}^{static}$ . Moreover, the lower bound on communication complexity of PRMT protocols are same against  $\mathcal{A}_{t_b}^{mobile}$ , as well as  $\mathcal{A}_{t_b}^{static}$ . Furthermore, the upper bound on communication complexity of PRMT protocols are also same against  $\mathcal{A}_{t_b}^{mobile}$ , as well as  $\mathcal{A}_{t_b}^{static}$ <sup>1</sup>. Interestingly, we show that it is not the case against mixed adversary. Specifically, in this chapter, we show that PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$  is possible iff PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$  is possible. That is, *mobility of the adversary has no effect on the POSSIBILITY of PRMT protocols tolerating mixed adversary*. However, the communication complexity required by PRMT protocols against  $\mathcal{A}_{(t_b, t_f)}^{mobile}$  is more in comparison to  $\mathcal{A}_{(t_b, t_f)}^{static}$ . That is, *mobility of the adversary has effect on the OPTIMALITY of PRMT protocols tolerating mixed adversary*. This brings forth the power of mobility in the context of mixed adversary. Our results in this chapter completely settle the issue of POSSIBILITY, FEASIBILITY and OPTIMALITY of PRMT protocols in undirected synchronous network, tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ . To the best of our knowledge, this is the first attempt in the literature of PRMT protocols.

We now begin with the specification of the network model used in this chapter.

### 6.1 Network Model and Adversary Settings

The network model used here is similar to the one given in section 4.1. That is, there are  $n$  bi-directional synchronous wires  $w_1, \dots, w_n$  between  $\mathbf{S}$  and  $\mathbf{R}$ . However, instead of  $\mathcal{A}_{t_b}^{mobile}$ , we assume the presence of a computationally unbounded mobile mixed adversary  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ . The adversary  $\mathcal{A}_{(t_b, t_f)}^{mobile}$  controls **different** set of  $t_b$  and  $t_f$  wires (among  $n$  wires), in Byzantine and fail-stop fashion respectively, in different phases

---

<sup>1</sup>Here there is a slight exception that against  $\mathcal{A}_{t_b}^{static}$ , we can design communication optimal PRMT protocol if  $\ell = \Theta(n^2)$  (see Fig. 3.5). On the other hand, against  $\mathcal{A}_{t_b}^{mobile}$ , we can design communication optimal PRMT protocol if  $\ell = \Theta(n^3)$  (see Fig. 4.9).

of a protocol. Hence if a wire is corrupted by the adversary in Byzantine/fail-stop fashion in  $i^{\text{th}}$  phase, then it is healed at the end of that phase. So a wire controlled by the adversary in  $i^{\text{th}}$  phase will be free from the influence of adversary in  $(i + 1)^{\text{th}}$  phase unless the adversary chooses the same wire to corrupt in  $(i + 1)^{\text{th}}$  phase as well. Though  $\mathcal{A}_{(t_b, t_f)}^{\text{mobile}}$  controls different set of wires in different phases of the protocol, it does not allow the adversary to gain any information which has previously passed (in earlier phases of the protocol) through the wires under its control in current phase (unless the wires were under the control of the adversary in earlier phases also). This is because the wires (and hence the nodes along these wires) erase all the local information from their memory at the end of each phase.

In the next section, we give the characterization of PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{mobile}}$ . We also derive the lower bound on the communication complexity of PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{mobile}}$ .

## 6.2 Characterization and Lower Bound on Communication Complexity

The necessary and sufficient condition for PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{mobile}}$  is given by the following theorem:

**Theorem 6.1** *PRMT over an undirected synchronous network  $\mathcal{N}$  tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{mobile}}$  is possible iff  $\mathcal{N}$  is  $(2t_b + t_f + 1)$ - $(\mathbf{S}, \mathbf{R})$ -connected.*

PROOF: From Theorem 5.1, PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{static}}$  is possible only if  $\mathcal{N}$  is  $(2t_b + t_f + 1)$ - $(\mathbf{S}, \mathbf{R})$ -connected. Since  $\mathcal{A}_{(t_b, t_f)}^{\text{mobile}}$  is more stronger than  $\mathcal{A}_{(t_b, t_f)}^{\text{static}}$ , obviously for PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{mobile}}$ , we require  $\mathcal{N}$  to be  $(2t_b + t_f + 1)$ - $(\mathbf{S}, \mathbf{R})$ -connected. This completes the necessity proof.

On the other hand, let  $\mathcal{N}$  be  $(2t_b + t_f + 1)$ - $(\mathbf{S}, \mathbf{R})$ -connected. That is, there exists at least  $n = 2t_b + t_f + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ . Then to reliably send a message  $m^{\mathbf{S}}$ , the sender  $\mathbf{S}$  can simply broadcast  $m^{\mathbf{S}}$  to  $\mathbf{R}$  over the  $n$  wires. It is easy to see that  $\mathbf{R}$  will correctly receive  $m^{\mathbf{S}}$  by taking majority among the received values. This completes the sufficiency proof.  $\square$

The lower bound on the communication complexity of PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{mobile}}$  is given by the following theorem:

**Theorem 6.2** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + t_f + 1$  wires and let  $r \geq 2$ . Then any  $r$ -phase<sup>2</sup> PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{mobile}}$ , must communicate  $\Omega\left(\frac{n\ell}{n-(t_b+t_f)}\right)$  field elements in order to transmit a message containing  $\ell$  field elements.*

PROOF: To prove the theorem, we first observe the communication pattern of any multi phase PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{mobile}}$  and show that the communication complexity of any multi phase PRMT protocol is not less than the communication complexity of a special type of single phase PRMT protocol. We then derive the lower bound on the communication complexity of this special type of single phase PRMT. More specifically,

---

<sup>2</sup>Any single phase PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{static}}$  will also work against  $\mathcal{A}_{(t_b, t_f)}^{\text{mobile}}$ . Hence the lower bound on the communication complexity of single phase PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{static}}$  as given in Theorem 5.5 will also hold against  $\mathcal{A}_{(t_b, t_f)}^{\text{mobile}}$ .

the proof of Theorem 6.2 follows from Lemma 6.3 and Lemma 6.11, which are proved below.

**Lemma 6.3** *The communication complexity of any multi-phase PRMT protocol to send a message against a mobile adversary corrupting up to  $B(\leq t_b)$  and  $F(\leq t_f)$  wires in Byzantine and fail-stop manner respectively (in each phase of the protocol) is not less than the share complexity (sum of the length of the shares) of distributing  $n$  shares for the message such that any set of  $n - B - F$  shares has full information about the message.*

To prove the above lemma, we begin with defining a weaker version of single-phase PRMT called PRMT with Error Detection (PRMTED). We then prove the equivalence between the communication complexity of PRMTED protocol to send message  $\mathbf{M}$  and the share complexity (sum of the length of all shares) of distributing  $n$  shares for  $\mathbf{M}$ , such that any set of  $n - B - F$  shares has full information about  $\mathbf{M}$  (see Claim 6.5). We then show that the communication complexity of any multiphase PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$  is at least equal to the communication complexity of single phase PRMTED (see Claim 6.10). These two equivalence will prove the desired equivalence as stated in Lemma 6.3. We begin with the definition of PRMTED protocol.

**Definition 6.4 (PRMTED)** *A single phase PRMT protocol is called PRMTED if it satisfies the following conditions:*

1. *If the adversary is passive throughout the protocol then  $\mathbf{R}$  correctly receives the message sent by  $\mathbf{S}$ .*
2. *If the adversary changes information over  $B$  wires ( $B \leq t_b$ ), then  $\mathbf{R}$  detects it, and aborts.*
3. *If adversary blocks  $F \leq t_f$  wires, without doing any other corruption, then  $\mathbf{R}$  recovers the message correctly. Else if adversary blocks more than  $t_f$  wires or does some other corruption (or both), then  $\mathbf{R}$  aborts.*

Observe that PRMTED is a strictly weaker version of PRMT because a PRMT protocol not only detects errors but also corrects them. We next show that the properties of PRMTED protocol for sending a message  $\mathbf{M}$  is equivalent to the problem of distributing  $n$  shares of  $\mathbf{M}$ , such that any set of  $n - B - F$  shares has full information about  $\mathbf{M}$ .

**Claim 6.5** *Let  $\Pi$  be a PRMTED protocol tolerating an adversary that can corrupt any  $B$  and  $F$  wires connecting  $\mathbf{S}$  and  $\mathbf{R}$  in Byzantine and fail-stop manner respectively. In an execution of  $\Pi$  for sending a message  $\mathbf{M}$ , the data  $s_i, i = 1, \dots, n$  sent by the  $\mathbf{S}$  along wires  $w_i$  form  $n$  shares for  $\mathbf{M}$ , such that any set of  $n - B - F$  shares has full information about  $\mathbf{M}$ .*

PROOF: We show that any set of  $n - B - F$  shares has full information about  $\mathbf{M}$ . The proof is by contradiction. For a set of wires  $A$ , let  $Message(\mathbf{M}, A)$  denote the set of messages sent along the wires in  $A$  during the execution of a PRMTED protocol to send  $\mathbf{M}$ . Now for any set of  $C$  wires with  $|C| \geq n - B - F$ ,  $Message(\mathbf{M}, C)$  should uniquely determine the message  $\mathbf{M}$ . If not, then there exists another message  $\mathbf{M}'$  such that  $Message(\mathbf{M}, C) = Message(\mathbf{M}', C)$ . By definition, the adversary can block all the messages sent along the  $F$  wires not in  $C$  and change the messages along  $B$  wires not

in  $C$ , such that the set of all messages received by  $\mathbf{R}$  is identical to  $Message(\mathbf{M}', C)$ . In this case,  $\mathbf{R}$  receives the message  $M'$ , while  $\mathbf{S}$  sent  $M$ . This is a contradiction since  $\mathbf{R}$  must detect that there has been a corruption.  $\square$

The above claim implies that the communication complexity of PRMTED protocol to send  $\mathbf{M}$  is same as the share complexity (sum of the length of all shares) of distributing  $n$  shares for  $\mathbf{M}$ , such that any set of  $n - B - F$  shares has full information about  $\mathbf{M}$ . Now we step forward to show that the communication complexity of PRMTED protocol is the lower bound on the communication complexity of any multiphase PRMT protocol against  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ . Before that we take a closer look at the execution of any multi-phase PRMT protocol against  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ .

In any multiphase PRMT protocol,  $\mathbf{S}$  and  $\mathbf{R}$  can be modelled as polynomial time Turing machines with access to a random tape. The number of random bits used by the  $\mathbf{S}$  and  $\mathbf{R}$  are bounded by a polynomial  $q(n)$ . Let  $r_1, r_2 \in \{0, 1\}^{q(n)}$  denote the contents of the random tapes of  $\mathbf{S}$  and  $\mathbf{R}$  respectively. The message  $\mathbf{M}$  is an element from the set  $\{0, 1\}^{p(n)}$ , where  $p(n)$  is a polynomial. A transcript for an execution of a multiphase PRMT protocol  $\Pi$  is the concatenation of all the messages sent by  $\mathbf{S}$  and  $\mathbf{R}$  along all the wires.

**Definition 6.6 (Passive Transcript)** *A passive transcript  $\mathcal{T}(\Pi, \mathbf{M}, r_1, r_2)$  is a transcript for the execution of the multiphase protocol  $\Pi$  with  $\mathbf{M}$  as the message to be sent,  $r_1, r_2$  as the contents of the random tapes of sender  $\mathbf{S}$  and the receiver  $\mathbf{R}$  and the adversary  $\mathcal{A}_{(t_b, t_f)}^{mobile}$  remaining passive throughout the execution of  $\Pi$ . Let  $\mathcal{T}(\Pi, \mathbf{M}, r_1, r_2, w_i)$  denote the passive transcript restricted to messages exchanged along the wire  $w_i$ . When  $\Pi, \mathbf{M}, r_1, r_2$  are obvious from the context, we drop them and denote the passive transcript restricted to a wire  $w_i$  by  $\mathcal{T}_{w_i}$ . Similarly,  $\mathcal{T}_C$  denotes the passive transcript  $\mathcal{T}$  restricted to a set of wires  $C$ .*

Given  $(\mathbf{M}, r_1, r_2)$  it is possible for  $\mathbf{S}$  to compute the passive transcript  $\mathcal{T}(\Pi, \mathbf{M}, r_1, r_2)$  by simulating  $\mathbf{R}$  with random tape  $r_2$ . Similarly given  $(\mathbf{M}, r_1, r_2)$ ,  $\mathbf{R}$  can compute  $\mathcal{T}(\Pi, \mathbf{M}, r_1, r_2)$  by simulating  $\mathbf{S}$  with  $r_1$ . Note that although  $\mathbf{S}$  and  $\mathbf{R}$  require both  $r_1, r_2$  to generate the transcript  $\mathcal{T}$ , receiver  $\mathbf{R}$  requires only  $r_2$  in order to obtain the message  $\mathbf{M}$  from the transcript  $\mathcal{T}$ . This is clear since  $\mathbf{R}$  does not have access to  $r_1$  during the execution of  $\Pi$  but still can retrieve the message  $\mathbf{M}$  from the messages exchanged.

**Definition 6.7 (Valid Fault Free Transcript)** *A passive transcript  $\mathcal{T}_C$ , with  $n - F \leq |C| \leq n$  is said to be a valid fault-free transcript with respect to  $\mathbf{R}$  if there exists random string  $r_2$  and message  $\mathbf{M}$  such that protocol  $\Pi$  at  $\mathbf{R}$  with  $r_2$  as the contents of the random tape and  $\mathcal{T}_C$  as the messages exchanged, terminates by outputting the message  $\mathbf{M}$ .*

**Definition 6.8 (Adversely Closed Transcript)** *Two passive transcripts  $\mathcal{T}_C$  and  $\mathcal{T}'_C$ , where  $n - F \leq C \leq n$  are said to be adversely close if the two transcripts differ only on a set of wires  $A$  such that  $|A| \leq B + (|C| - (n - F))$ . Formally  $|\{w_i | \mathcal{T}_{w_i} \neq \mathcal{T}'_{w_i}\}| \leq B + (|C| - (n - F))$ .*

We next prove the following claim:

**Claim 6.9** *Two valid fault-free transcripts  $\mathcal{T}_C(\Pi, \mathbf{M}, r_1, r_2)$  and  $\mathcal{T}_C(\Pi, \mathbf{M}', r'_1, r'_2)$  with two different message inputs  $\mathbf{M}, \mathbf{M}'$ , cannot be adversely close to each other, where  $n - F \leq C \leq n$ .*

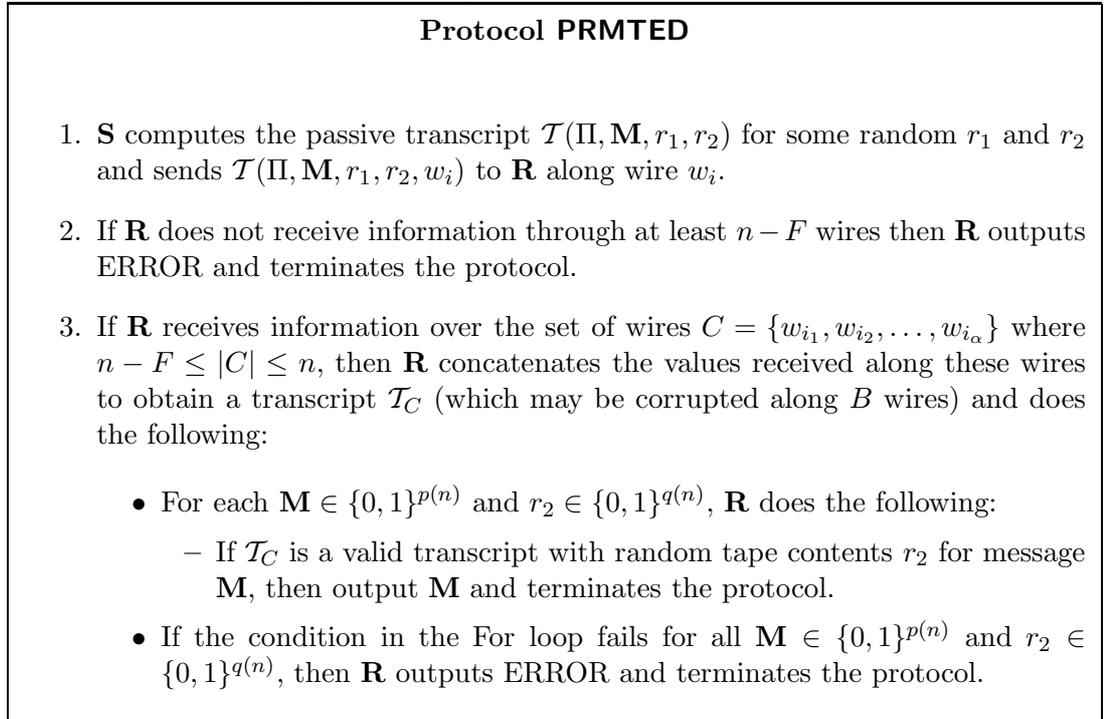
PROOF: Suppose two valid fault-free transcripts  $\mathcal{T}_C(\Pi, \mathbf{M}, r_1, r_2)$  and  $\mathcal{T}_C(\Pi, \mathbf{M}', r'_1, r'_2)$  are adversely close. This implies that there is a set of wires  $A$ , where  $|A| \leq B + (|C| - (n - F))$ , such that the two transcripts differ only on messages sent along the wires in  $A$ . Without loss of generality, assume that the last  $B + (|C| - (n - F))$  wires belong to  $A$  with  $A = X \circ Y$  where  $|X| = B$  and  $|Y| = (|C| - (n - F))$ . Consider the following two executions of  $\Pi$  where the contents of  $\mathbf{S}$ 's and  $\mathbf{R}$ 's random tapes are  $r_1, r_2$  respectively

- $\mathbf{S}$  wants to send  $\mathbf{M}$ .  $\mathbf{S}$  and  $\mathbf{R}$  executes  $\Pi$  while the adversary stop the wires in  $Y$  to deliver any message. As  $\mathcal{T}_{C-Y}(\Pi, \mathbf{M}, r_1, r_2)$  is a valid transcript with respect to  $\mathbf{M}$ ,  $\mathbf{R}$  terminates with output  $\mathbf{M}$ .
- $\mathbf{S}$  wants to send  $\mathbf{M}$ .  $\mathbf{S}$  and  $\mathbf{R}$  executes  $\Pi$ . The adversary blocks messages over  $Y$  and changes the messages along wires in  $X$  such that the view of  $\mathbf{S}$  is  $\mathcal{T}_{C-Y}(\Pi, \mathbf{M}, r_1, r_2)$  but the view of  $\mathbf{R}$  is  $\mathcal{T}_{C-Y}(\Pi, \mathbf{M}', r'_1, r'_2)$ . Since  $\mathcal{T}_{C-Y}(\Pi, \mathbf{M}', r'_1, r'_2)$  is a valid transcript with respect to  $\mathbf{M}'$ ,  $\mathbf{R}$  will terminate with output  $\mathbf{M}'$ .

The two scenarios differ only in the adversarial behavior and in the contents of  $\mathbf{R}$ 's random tape. In both the scenarios  $\mathbf{S}$  wanted to send message  $\mathbf{M}$ . But the message received by receiver  $\mathbf{R}$  in the second case is an incorrect message  $\mathbf{M}'$ . This is a contradiction because  $\Pi$  is a PRMT protocol.  $\square$

Till now, we have shown that a passive transcript over at least  $n - B - F$  wires allows  $\mathbf{R}$  to output  $\mathbf{M}$  correctly. We now show how to reduce a multiphase PRMT protocol into a single phase PRMTED protocol. Specifically, consider the protocol PRMTED, given in Fig. 6.1.

Figure 6.1: Single Phase PRMTED Protocol



We now make the following claim regarding protocol PRMTED.

**Claim 6.10** *The Communication complexity of any multiphase PRMT protocol  $\Pi$  against  $\mathcal{A}_{(t_b, t_f)}^{mobile}$  is at least equal to the communication complexity of PRMTED protocol. Moreover, protocol PRMTED satisfies the properties given in Definition 6.4.*

PROOF: Let  $\Pi$  be any multi phase PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$  and let  $\Pi^{passive}$  denote an execution of  $\Pi$ , where the adversary does only eavesdropping and does no other type of corruption during the complete execution. It is easy to see that the communication complexity of  $\Pi^{passive}$  is trivially a lower bound on the communication complexity of any multi phase PRMT protocol (where the adversary may do other types of corruptions, in addition to eavesdropping). We now show that the communication complexity of  $\Pi^{passive}$  is same as the communication complexity of PRMTED protocol. Once we do this, then the communication complexity of PRMTED protocol is a trivial lower bound on the communication complexity of any multi phase PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ .

In PRMTED,  $\mathbf{S}$  assumes its random tape contains  $r_1$  and random tape of  $\mathbf{R}$  contains  $r_2$ .  $\mathbf{S}$  also assumes that in  $\Pi$ , the adversary will only do eavesdropping and no other type of corruption and generates the passive transcript  $\mathcal{T}(\Pi, \mathbf{M}, r_1, r_2)$ . As explained earlier,  $\mathbf{S}$  can do so by simulating  $\mathbf{R}$ , assuming the content of  $\mathbf{R}$ 's random tape to be  $r_2$ . However, note that  $\mathbf{R}$  neither knows  $m$ , nor  $r_1, r_2$ , which  $\mathbf{S}$  has used for generating  $\mathcal{T}$ .  $\mathbf{S}$  then communicates  $\mathcal{T}$  to  $\mathbf{R}$  by sending the components of  $\mathcal{T}$  restricted to wire  $w_i$ , along  $w_i$ . It is easy to see that the cost of communicating such a transcript by PRMTED is same as the communication complexity of  $\Pi^{passive}$ .

The messages sent along wire  $w_i$  in protocol PRMTED is the concatenation of the messages that would have been exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  along  $w_i$  in  $\Pi^{passive}$ . From Claim 6.9, we know that valid transcripts of two different messages cannot be adversely close to each other. So irrespective of the actions of the adversary, the transcript received by  $\mathbf{R}$  cannot be a valid transcript for any message other than  $\mathbf{M}$  for any value of  $r_2$ . Hence if  $\mathbf{R}$  outputs a message  $\mathbf{M}$  then it is the same message sent by  $\mathbf{S}$ . Thus protocol PRMTED satisfies the properties given in Definition 6.4.  $\square$

Claim 6.5, along with Claim 6.10 completes the proof of Lemma 6.3. Till now, we have shown that the communication complexity of PRMTED protocol is the lower bound on the communication complexity of any multi phase PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ . Moreover, the communication complexity of PRMTED protocol to send a message  $\mathbf{M}$  is same as the share complexity of distributing  $n$  shares for the message  $\mathbf{M}$ , such that any set of  $n - (F + B)$  shares has full information about the message. All these facts implies that share complexity of such a distribution gives the lower bound on the communication complexity of any multi phase PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ . We now proceed to derive the share complexity of such a distribution scheme.

**Lemma 6.11** *The share-complexity (that is sum of the length of all shares) of distributing  $n$  shares for a message, consisting of  $\ell$  elements from  $\mathbb{F}$ , such that any set of  $n - B - F$  shares has full information about the message, is  $\Omega\left(\frac{n\ell}{(n-B-F)}\right)$ .*

PROOF: To prove the lemma, we use entropy based arguments. Let  $X_i$  denote the  $i^{th}$  share. For any subset  $A \subseteq \{1, 2, \dots, n\}$ , let  $X_A$  denote the set of variables  $\{X_i | i \in A\}$ . Let  $\mathbf{M}$  be the message, selected uniformly at random from  $\mathbb{F}^\ell$ . Then the message  $\mathbf{M}$  and the shares  $X_i$  are random variables. Let  $H(X)$  for a random variable denote its entropy. Let  $H(X|Y)$  denote the entropy of  $X$  conditional on  $Y$ . The conditional entropy measures how much entropy a random variable  $X$  has remaining, if we have

already learned completely the value of the second random variable  $Y$  [22]. Since any set  $C$  of  $n - B - F$  shares has full information about  $\mathbf{M}$ , we have

$$H(\mathbf{M}|X_C) = 0$$

Since  $\mathbf{M}$  is chosen uniformly from  $\mathbb{F}^\ell$ , we have

$$H(\mathbf{M}) = \ell \tag{6.1}$$

From the chain rule of the entropy [22], for any two random variable  $X_1, X_2$ , we have  $H(X_1, X_2) = H(X_2) + H(X_1|X_2)$ . Substituting  $X_1 = \mathbf{M}$  and  $X_2 = X_C$ , we get

$$H(\mathbf{M}, X_C) = H(X_C) + H(\mathbf{M}|X_C)$$

From the properties of joint entropy [22], for any two variables  $X_1, X_2$ , we have  $H(X_1, X_2) \geq H(X_1)$  and  $H(X_1, X_2) \geq H(X_2)$ . Thus,  $H(\mathbf{M}, X_C) \geq H(\mathbf{M})$ . Substituting in the above equation, we get

$$\begin{aligned} H(\mathbf{M}) &\leq H(X_C) + H(\mathbf{M}|X_C) \\ &\leq H(X_C) + 0 \text{ because } \mathbf{M} \text{ can be known completely from } X_C \end{aligned}$$

Consequently,  $H(\mathbf{M}) \leq H(X_C)$ . Therefore for any set  $C$  of cardinality  $n - B - F$ , we have

$$H(X_C) \geq H(\mathbf{M}) \Rightarrow \sum_{i \in C} H(X_i) \geq H(\mathbf{M})$$

Since there are  $\binom{n}{n-B-F}$  possible subsets of cardinality  $n - B - F$ , summing the above equation over all possible subsets of cardinality  $n - B - F$  we get

$$\sum_C \sum_{i \in C} H(X_i) \geq \binom{n}{n-B-F} H(\mathbf{M})$$

Now in all the possible  $\binom{n}{n-B-F}$  subsets of size  $n - B - F$ , each of the term  $H(X_i)$  appears  $\binom{n-1}{n-B-F-1}$  times. So

$$\binom{n-1}{n-B-F-1} \sum_{i=1}^n H(X_i) \geq \binom{n}{n-B-F} H(\mathbf{M}) \Rightarrow \sum_{i=1}^n H(X_i) \geq \frac{n}{n-B-F} H(\mathbf{M})$$

Since  $H(\mathbf{M}) = \ell$ , we get

$$\sum_{i=1}^n H(X_i) \geq \frac{n\ell}{n-B-F}$$

Thus the share-complexity for any  $\mathbf{M} \in \mathbb{F}^\ell$  is  $\Omega\left(\frac{n\ell}{n-B-F}\right)$ .  $\square$

Since  $B \leq t_b$  and  $F \leq t_f$ ,  $\Omega\left(\frac{n\ell}{n-B-F}\right) = \Omega\left(\frac{n\ell}{n-(t_b+t_f)}\right)$ . Theorem 6.2 now follows from Lemma 6.3 and Lemma 6.11.  $\square$

### 6.2.1 PRMT Against $\mathcal{A}_{(t_b, t_f)}^{mobile}$ Requires More Communication Than PRMT Against $\mathcal{A}_{(t_b, t_f)}^{static}$

Suppose  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + t_f + 1$  wires, which are under the control of static mixed adversary  $\mathcal{A}_{(t_b, t_f)}^{static}$ . Then from Corollary 5.17.2, any PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ , must run for  $\Omega(\log(\frac{t_f}{t_b}))$  phases to reliably send  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements. Moreover, from Theorem 5.27, the bound on phase complexity is tight. This shows that in the presence of  $\mathcal{A}_{(t_b, t_f)}^{static}$ , it is possible to achieve reliability with constant factor overhead, where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + t_f + 1$  wires.

Now suppose that  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + t_f + 1$  wires, which are under the control of mobile mixed adversary  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ . Then from Theorem 6.2, any multiphase PRMT protocol against  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ , must communicate  $\Omega\left(\frac{n\ell}{t_b}\right)$  field elements to reliably send  $\ell$  field elements. However, since  $n = 2t_b + t_f + 1$ ,  $n$  need not be  $\Theta(t_b)$ . In fact, in the worst case,  $t_b$  may be constant. This implies that in the presence of  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ , it may not be possible at all to achieve reliability with constant factor overhead, irrespective of the number of phases, when  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + t_f + 1$  wires. This shows that any PRMT protocol against  $\mathcal{A}_{(t_b, t_f)}^{mobile}$  requires more communication complexity in comparison to its static counter part.

In the next section, we show that the bound on the communication complexity given in Theorem 6.2 is asymptotically tight. For this, we design a communication optimal PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ .

### 6.3 Communication Optimal PRMT Tolerating $\mathcal{A}_{(t_b, t_f)}^{mobile}$

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + t_f + 1$  wires, which are under the control of  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ . We then design a three phase PRMT protocol called 3-Optimal-PRMT-Mobile-Mixed, which reliably sends a message  $m^{\mathbf{S}}$  containing  $nt_b$  ( $t_b \geq 1$ ) field elements by communicating  $\mathcal{O}(n^2)$  field elements. If  $t_b = \Theta(n)$ , then  $nt_b = \Theta(n^2)$  and hence the protocol sends  $\Theta(n^2)$  field elements by communicating  $\mathcal{O}(n^2)$  field elements, thus achieving reliability with constant factor overhead.

**Remark 6.12** Notice that if  $t_b = 0$ , then  $n = t_f + 1$ . In this case, we can directly send any message of size  $\ell$  by broadcasting it over the  $n$  wires, incurring a communication cost of  $\mathcal{O}(n\ell)$ . Moreover, from Theorem 6.2, this will be optimal.

The idea of protocol 3-Optimal-PRMT-Mobile-Mixed is as follows: we create a win-win situation with the adversary. Specifically,  $\mathbf{S}$  divides the message into several blocks of size  $\frac{t_b}{2} + 1$  and tries to reliably send each block by encoding it into an RS code word of length  $n$ . If at most  $\frac{t_b}{2}$  Byzantine errors occur during the transmission, then  $\mathbf{R}$  will be able to correctly recover each block at the end of first phase itself. However, if more than  $\frac{t_b}{2}$  Byzantine errors occur, then  $\mathbf{R}$  will fail to recover the blocks, but will detect that more than  $\frac{t_b}{2}$  errors have occurred. So  $\mathbf{R}$  reliably sends back the received vector to  $\mathbf{S}$ , in which  $\mathbf{R}$  has detected more than  $\frac{t_b}{2}$  errors. However, since the adversary is mobile and may corrupt different set of wires during second phase, to reliably send back the received vector,  $\mathbf{R}$  broadcasts it to  $\mathbf{S}$ .  $\mathbf{S}$ , after local comparison, finds the identity of at least  $\frac{t_b}{2}$  wires, which delivered incorrect information during the first phase. During third phase,  $\mathbf{S}$  reliably sends the identity of these corrupted wires to  $\mathbf{R}$  by broadcasting. Now from the each vector received by  $\mathbf{R}$  during first phase,  $\mathbf{R}$  will

neglect the components received over the corrupted wires.  $\mathbf{R}$  will be then left with shortened vector and in each vector there will be at most  $\frac{t_b}{2}$  corrupted values, which  $\mathbf{R}$  can now easily correct by using RS decoding algorithm. Thus in short, second and third phase are used to identify at least  $\frac{t_b}{2}$  corrupted values in the vectors, which were received by  $\mathbf{R}$  during first phase. Once these corrupted values are removed, there will be sufficient redundancy in each received vector to recover the original blocks of  $m^{\mathbf{S}}$ . The protocol is formally given in Fig. 6.2.

Figure 6.2: A Three Phase Communication Optimal PRMT Protocol Tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ ,  $n = 2t_b + t_f + 1$ ,  $|m^{\mathbf{S}}| = nt_b$

**Protocol 3-Optimal-PRMT-Mobile-Mixed**

**Phase I: S to R:**

1.  $\mathbf{S}$  divides  $m^{\mathbf{S}}$  into blocks  $\mathbf{B}_1^{\mathbf{S}}, \dots, \mathbf{B}_z^{\mathbf{S}}$ , each containing  $1 + \frac{t_b}{2}$  field elements.
2. For  $j = 1, \dots, z$ ,  $\mathbf{S}$  computes the RS codeword  $C_j^{\mathbf{S}} = (c_{j1}^{\mathbf{S}}, \dots, c_{jn}^{\mathbf{S}})$  of length  $n$  corresponding to block  $\mathbf{B}_j^{\mathbf{S}}$ .
3. For  $i = 1, \dots, n$ ,  $\mathbf{S}$  sends  $i^{th}$  component of all codewords to  $\mathbf{R}$  over wire  $w_i$ .

**Phase II: R to S:**

1. Let  $\mathbf{R}$  receive information over wires  $w_{i_1}, \dots, w_{i_{N'}}$ , where  $n - t_f \leq N' \leq n$ .
2. For  $j = 1, \dots, z$ , let  $\mathbf{R}$  receive  $c_{ji_1}^{\mathbf{R}}, \dots, c_{ji_{N'}}^{\mathbf{R}}$  over wires  $w_{i_1}, \dots, w_{i_{N'}}$  respectively. Let  $C_j^{\mathbf{R}} = (c_{ji_1}^{\mathbf{R}}, \dots, c_{ji_{N'}}^{\mathbf{R}})$ .
3. For  $j = 1, \dots, z$ ,  $\mathbf{R}$  executes  $\text{RS-DEC}(N', C_j^{\mathbf{R}}, \frac{t_b}{2}, \frac{t_b}{2}, \frac{t_b}{2} + 1)$ .
4. If  $\exists j \in \{1, 2, \dots, z\}$ , such that  $\text{RS-DEC}(N', C_j^{\mathbf{R}}, \frac{t_b}{2}, \frac{t_b}{2}, \frac{t_b}{2} + 1)$  does not output any polynomial of degree  $\frac{t_b}{2}$ , then  $\mathbf{R}$  broadcasts  $C_j^{\mathbf{R}}$  and index  $j$  to  $\mathbf{S}$ .
5. Else for each  $C_j^{\mathbf{R}}, j = 1, \dots, z$ ,  $\text{RS-DEC}(N', C_j^{\mathbf{R}}, \frac{t_b}{2}, \frac{t_b}{2}, \frac{t_b}{2} + 1)$  outputs correct  $\mathbf{B}_j^{\mathbf{S}}$ .  $\mathbf{R}$  recovers  $m^{\mathbf{S}}$  by concatenating all  $\mathbf{B}_j^{\mathbf{S}}$ 's, broadcasts "TERMINATE" signal to  $\mathbf{S}$  and terminates the protocol.

If  $\mathbf{S}$  receives "TERMINATE" signal, then  $\mathbf{S}$  terminates the protocol. Else  $\mathbf{S}$  executes the third phase as follows:

**Phase III: S to R:**

1.  $\mathbf{S}$  correctly receives  $C_j^{\mathbf{R}}$  and index  $j$ .
2.  $\mathbf{S}$  locally compares  $C_j^{\mathbf{R}}$  with  $C_j^{\mathbf{S}}$  and identifies more than  $\frac{t_b}{2}$  wires which were Byzantine corrupted during **Phase I**.  $\mathbf{S}$  saves the identities of these wires in a list  $L_{fault}$ .
3.  $\mathbf{S}$  broadcasts to  $\mathbf{R}$  the list  $L_{fault}$  and terminates the protocol.

**Local Computation by R at the End of Phase III:**

1.  $\mathbf{R}$  correctly receives  $L_{fault}$  and identifies  $|L_{fault}| \geq \frac{t_b}{2} + 1$  wires, which delivered incorrect values during **Phase I**.
2. For  $j = 1, \dots, z$ ,  $\mathbf{R}$  removes  $c_{ji}^{\mathbf{R}}$  from  $C_j^{\mathbf{R}}$ , provided  $w_i \in L_{fault}$ .
3. For  $j = 1, \dots, z$ ,  $\mathbf{R}$  executes  $\text{RS-DEC}(N' - |L_{fault}|, C_j^{\mathbf{R}}, t_b - |L_{fault}|, 0, \frac{t_b}{2} + 1)$ .
4. For  $j = 1, \dots, z$ ,  $\text{RS-DEC}(N' - |L_{fault}|, C_j^{\mathbf{R}}, t_b - |L_{fault}|, 0, \frac{t_b}{2} + 1)$  outputs  $\mathbf{B}_j^{\mathbf{S}}$ .  $\mathbf{R}$  recovers  $m^{\mathbf{S}}$  by concatenating all  $\mathbf{B}_j^{\mathbf{S}}$ 's and terminates the protocol.

Our three phase protocol works in the presence of  $\mathcal{A}_{(t_b, t_f)}^{mobile}$  as in **Phase II** and **Phase III**,  $\mathbf{R}/\mathbf{S}$  uses only broadcast for reliably sending some information. Broadcast sends any information reliably even in the presence of  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ . We now prove the properties of protocol 3-Optimal-PRMT-Mobile-Mixed.

**Lemma 6.13 (Correctness)** *Protocol 3-Optimal-PRMT-Mobile-Mixed correctly delivers the message  $m^{\mathbf{S}}$  in at most three phases, tolerating  $\mathcal{A}_{(t_b, t_f)}^{\text{mobile}}$ .*

PROOF: We prove the theorem for the worst case, when  $\mathbf{R}$  receives information over  $N' = n - t_f = 2t_b + 1$  wires during **Phase I**. Thus each received vector  $C_j^{\mathbf{R}}$  will be of length  $2t_b + 1$ . Each  $C_j^{\mathbf{R}}$  will differ from  $C_j^{\mathbf{S}}$  in at most  $t_b$  locations. Moreover, each  $C_j^{\mathbf{S}}$  is RS encoded using polynomial of degree  $\frac{t_b}{2}$ . Now by putting  $N' = 2t_b + 1, c = d = \frac{t_b}{2}$  and  $k = \frac{t_b}{2} + 1$  in the inequality of Theorem 2.19, we find that RS-DEC( $N', C_j^{\mathbf{R}}, \frac{t_b}{2}, \frac{t_b}{2}, \frac{t_b}{2} + 1$ ) will be able to correct  $\frac{t_b}{2}$  errors in  $C_j^{\mathbf{R}}$  and detect additional  $\frac{t_b}{2}$  errors (if any) in  $C_j^{\mathbf{R}}$ . Now there are two possible cases:

1. **At most  $\frac{t_b}{2}$  Byzantine Errors are Present in Each  $C_j^{\mathbf{R}}$ :** In this case, RS-DEC will correct all these errors and will detect no additional errors. Thus RS-DEC will correctly output each  $\mathbf{B}_j^{\mathbf{S}}$ . Moreover,  $\mathbf{R}$  will know that  $\mathbf{B}_j^{\mathbf{S}}$  which is output by RS-DEC is correct. This is because in this case, the error correction plus detection capability of RS-DEC is equal to the maximum number of errors that can happen (i.e  $c + d = t_b$ ). So  $\mathbf{R}$  will correctly recover  $m^{\mathbf{S}}$  and will terminate the protocol by broadcasting "TERMINATE" signal <sup>3</sup>.  $\mathbf{S}$  on receiving this signal will conclude that  $\mathbf{R}$  has recovered  $m^{\mathbf{S}}$  and hence will terminate the protocol. So, in this case, the protocol will terminate in *two* phases.
2. **More than  $\frac{t_b}{2}$  Errors are Present in Some  $C_j^{\mathbf{R}}$ :** In this case, RS-DEC will correct  $\frac{t_b}{2}$  errors in  $C_j^{\mathbf{R}}$  and will detect the remaining errors (which can be at most  $\frac{t_b}{2}$ ). Since RS-DEC can only detect the remaining errors and has no capability of correcting them, it will not output anything. This will indicate to  $\mathbf{R}$  that more than  $\frac{t_b}{2}$  errors are present in  $C_j^{\mathbf{R}}$ . However,  $\mathbf{R}$  has no means to know the identity of the corrupted wires, who delivered those corrupted components of  $C_j^{\mathbf{R}}$  <sup>4</sup>. To know their identities,  $\mathbf{R}$  broadcasts  $C_j^{\mathbf{R}}$  to  $\mathbf{S}$ .

Upon receiving  $C_j^{\mathbf{R}}$  and locally comparing it with  $C_j^{\mathbf{S}}$ ,  $\mathbf{S}$  will identify at least  $\frac{t_b}{2} + 1$  Byzantine corrupted wires who had sent incorrect values during **Phase I**.  $\mathbf{S}$  saves their identities in a list  $L_{\text{fault}}$ .  $\mathbf{S}$  then broadcasts  $L_{\text{fault}}$  to  $\mathbf{R}$ . On receiving  $L_{\text{fault}}$ ,  $\mathbf{R}$  removes all the components of each  $C_j^{\mathbf{R}}$  received over the wires in  $L_{\text{fault}}$ . Thus each  $C_j^{\mathbf{R}}$  will now contain  $2t_b + 1 - |L_{\text{fault}}|$  values, out of which at most  $t_b - |L_{\text{fault}}| < \frac{t_b}{2}$  could be corrupted. Now putting  $N' = N' - |L_{\text{fault}}| = 2t_b + 1 - |L_{\text{fault}}|, c = t_b - |L_{\text{fault}}|, d = 0$  and  $k = \frac{t_b}{2} + 1$  in the inequality of Theorem 2.19, we find that RS-DEC will be able to correct all  $t_b - |L_{\text{fault}}|$  errors present in each  $C_j^{\mathbf{R}}$  and will correctly output  $\mathbf{B}_j^{\mathbf{S}}$ . Moreover, since  $\mathbf{R}$  now knows that at most  $t_b - |L_{\text{fault}}| < \frac{t_b}{2}$  errors are present in each  $C_j^{\mathbf{R}}$ ,  $\mathbf{R}$  concludes that output  $\mathbf{B}_j^{\mathbf{S}}$  is correct. Thus by combining all  $\mathbf{B}_j^{\mathbf{S}}$ 's,  $\mathbf{R}$  will recover  $m^{\mathbf{S}}$  correctly. So, in this case, the protocol will terminate in *three* phases.

This completes the proof of the correctness. □

**Lemma 6.14 (Communication Complexity)** *Protocol 3-Optimal-PRMT-Mobile-Mixed sends  $m^{\mathbf{S}}$  containing  $nt_b$  field elements by communicating  $\mathcal{O}(n^2)$  field elements.*

PROOF: During **Phase I**,  $\mathbf{S}$  sends an RS codeword of length  $n$  for each  $\mathbf{B}_j^{\mathbf{S}}$ , where size of  $\mathbf{B}_j^{\mathbf{S}}$  is  $1 + \frac{t_b}{2}$ . So the total communication cost of **Phase I** is  $\mathcal{O}\left(\frac{|m^{\mathbf{S}}|}{\frac{t_b}{2}} * n\right) = \mathcal{O}(n^2)$ ,

<sup>3</sup>This case is similar to Property 2.22.

<sup>4</sup>This case is similar to Property 2.23.

as  $|m^{\mathbf{S}}| = nt_b$ . In the second phase,  $\mathbf{R}$  may either broadcast "TERMINATE" signal or a vector  $C_j^{\mathbf{R}}$  of length  $N'$ . So in the worst case, the communication cost of **Phase II** is  $\mathcal{O}(n^2)$ . If **Phase III** is executed, then  $\mathbf{S}$  broadcasts  $L_{fault}$ , where  $\frac{t_b}{2} + 1 \leq |L_{fault}| \leq t_b$ . This incurs a communication cost of  $\mathcal{O}(nt_b)$ . Thus the overall communication cost of 3-Optimal-PRMT-Mobile-Mixed is  $\mathcal{O}(n^2)$ .  $\square$

**Theorem 6.15** *Protocol 3-Optimal-PRMT-Mobile-Mixed is a communication optimal PRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ .*

PROOF: From Theorem 6.2, we find that any three phase PRMT protocol over  $n = 2t_b + t_f + 1$  wires must communicate  $\Omega(n^2)$  field elements to reliably send a message containing  $nt_b$  field elements against  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ . From Lemma 6.14, the communication complexity of protocol 3-Optimal-PRMT-Mobile-Mixed is  $\mathcal{O}(n^2)$ . Hence it is communication optimal.  $\square$

## 6.4 Concluding Remarks and Open Problems

In this chapter, we have studied the issues related to POSSIBILITY, FEASIBILITY and OPTIMALITY of PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ . Our results show that mobility of the mixed adversary has no effect on the POSSIBILITY of PRMT. However, mobility of the mixed adversary does affect the OPTIMALITY of PRMT protocols. The results for PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$  are summarized in Fig. 6.3.

Figure 6.3: Summary of the Results for PRMT in Undirected Synchronous Network Tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$

Number of Phases ( $r$ )	Connectivity Requirement ( $n$ )	Lower Bound on Communication Complexity	Upper Bound
$r \leq 1$	$n \geq 2t_b + t_f + 1$ Theorem 6.1	$\Omega\left(\frac{n\ell}{n-(2t_b+t_f)}\right)$ Theorem 5.5	Broadcast protocol: $n = 2t_b + t_f + 1$ , Communication complexity = $\mathcal{O}(n\ell)$
$r \geq 2$	$n \geq 2t_b + t_f + 1$ Theorem 6.1	$\Omega\left(\frac{n\ell}{n-(t_b+t_f)}\right)$ Theorem 6.2	Protocol 3-Optimal-PRMT-Mobile-Mixed: $n = 2t_b + t_f + 1, \ell = \Theta(nt_b)$ Communication complexity = $\mathcal{O}\left(\frac{n\ell}{t_b}\right)$

From Fig. 6.3, we find that protocol 3-Optimal-PRMT-Mobile-Mixed is communication optimal only if the message contains  $\ell = \Theta(nt_b)$  field elements. This leads to the following open question:

**Open Problem 5** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + t_f + 1$  wires. Then does there exist a multiphase (more than one phase) PRMT protocol which reliably sends a message containing  $\ell$  field elements, by communicating  $\mathcal{O}\left(\frac{n\ell}{t_b}\right)$  field elements, tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ , for any value of  $\ell$ ?*

## Chapter 7

# PRMT in Directed Networks Tolerating Static Byzantine Adversary

Till now, all the results that we discussed are in the undirected network model, where the communication link between any two nodes in the network is bi-directional. However, in practice, not every communication channel may admit bi-directional communication. For instance, a base-station may communicate to even a far-off hand-held device but the other way round communication may not be possible. In such a scenario, it is more appropriate to model the underlying network as a directed graph. Motivated by this, Desmedt et al. [24, 87] introduced the problem of PRMT, PSMT, SRMT and SSMT in directed network. The necessary and sufficient condition for PRMT in undirected graph tolerating  $\mathcal{A}_{t_b}^{static}$  as given in Theorem 3.2 will also hold for undirected graphs. However, to the best of our knowledge, nothing is known regarding the OPTIMALITY of PRMT in directed network. We completely resolve this issue in this chapter.

We now give the formal description of the directed network model, which is used in this chapter.

### 7.1 Directed Network Model

We assume that the underlying network is a connected, synchronous network represented by a *directed* graph where  $\mathbf{S}$  and  $\mathbf{R}$  are two *non-adjacent* nodes of the graph. All the arcs in the network are reliable and secure but the nodes can be corrupted. The intermediate nodes between  $\mathbf{S}$  and  $\mathbf{R}$  are oblivious, message passing nodes and they do no computation of their own. Their only task is to pass information from their predecessor node to their successor node.

We assume the presence of a static, threshold adversary  $\mathcal{A}_{t_b}^{static}$ , having *unbounded computing power*, who can corrupt any set of  $t_b$  nodes in the graph (excluding  $\mathbf{S}$  and  $\mathbf{R}$ ) in Byzantine fashion. Following the approach of [24], we abstract away the network and assume that  $\mathbf{S}$  and  $\mathbf{R}$  are connected by node disjoint paths, also called as *wires*, which are directed either from  $\mathbf{S}$  to  $\mathbf{R}$  or vice-versa. More specifically, we assume that there are  $n$  wires from  $\mathbf{S}$  to  $\mathbf{R}$ , denoted by  $f_1, \dots, f_n$  and  $u$  wires from  $\mathbf{R}$  to  $\mathbf{S}$ , denoted by  $b_1, \dots, b_u$ . Moreover, the wires from  $\mathbf{S}$  to  $\mathbf{R}$  are node disjoint from the wires which are directed from  $\mathbf{R}$  to  $\mathbf{S}$ . The  $n$  wires from  $\mathbf{S}$  to  $\mathbf{R}$  are also called as *top band*, while the  $u$  wires from  $\mathbf{R}$  to  $\mathbf{S}$  are called as *bottom band*. The adversary  $\mathcal{A}_{t_b}^{static}$  can control

at most  $t_b$  wires out of these  $n + u$  wires in Byzantine fashion.

## 7.2 Characterization of Communication Optimal PRMT in Directed Network

We first begin with the following theorem:

**Theorem 7.1** *PRMT between  $\mathbf{S}$  and  $\mathbf{R}$  in an undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff there exists  $n \geq 2t_b + 1$  wires in the top band.*

PROOF: The necessity follows from Theorem 3.2. For sufficiency, we can use broadcast protocol.  $\square$

The communication complexity of the broadcast protocol is  $\mathcal{O}(n\ell)$ . Now an interesting question here is whether we can further reduce the communication complexity of PRMT protocol in directed network. Any PRMT protocol which reliably sends a message containing  $\ell$  field elements against  $\mathcal{A}_{t_b}^{static}$  has a trivial lower bound of  $\Omega(\ell)$  on communication complexity. This is because any PRMT protocol has to reliably send the message. Thus any PRMT protocol which reliably sends a message containing  $\ell$  field elements and has a communication complexity of  $\mathcal{O}(\ell)$  field elements will be communication optimal PRMT protocol. Indeed, in Chapter 3, we have designed a three phase PRMT protocol called 3-Optimal-PRMT-Static-Byzantine, which reliably sends a message containing  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements in undirected network, tolerating  $\mathcal{A}_{t_b}^{static}$ . We now show that something similar can be also done in directed network.

Before proceeding further, we stop for a moment and compare the directed network model with the undirected network model. In undirected network model tolerating  $\mathcal{A}_{t_b}^{static}$ , we assume that there exists  $n \geq 2t_b + 1$  bi-directional wires between  $\mathbf{S}$  and  $\mathbf{R}$ , of which at most  $t_b$  wires could be under the control of  $\mathcal{A}_{t_b}^{static}$ . Thus  $\mathbf{S}$  and  $\mathbf{R}$  can always do reliable communication between them by broadcasting information over the  $n$  wires. This allows us to do error detection and correction and design a three phase communication optimal PRMT protocol, which allows to send  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements (see protocol 3-Optimal-PRMT-Static-Byzantine).

On the other hand, in directed network model, there are  $n$  wires in the top band and  $u$  wires in the bottom band, where the wires in the top band are completely disjoint from the wires in the bottom band. From Theorem 7.1, for any PRMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ ,  $n \geq 2t_b + 1$ . So  $\mathbf{S}$  can always reliably send any information to  $\mathbf{R}$ . But  $u$  can be anything. In fact,  $u$  may be less than or equal to  $t_b$  and it may happen that the entire bottom band is corrupted. So in any multiphase PRMT protocol in directed network,  $\mathbf{R}$  may not always be able to reliably send feedback. This is in contrast to the case of undirected network, where  $\mathbf{R}$  can always do so by simply broadcasting the feedback. It is this inherent difficulty, which makes the task of designing communication optimal PRMT protocol little difficult in the case of directed network and hence call for some new techniques. In the next theorem, we first characterize the digraphs, over which communication optimal PRMT protocol is possible. To be more clear, we answer the following question:

*what are the necessary and sufficient structural conditions that the underlying directed graph should satisfy for the possibility of communication optimal PRMT protocol, which sends a message consisting of  $\ell$  elements from  $\mathbb{F}$ , by communicating  $\mathcal{O}(\ell)$  field elements, tolerating  $\mathcal{A}_{t_b}^{static}$ ?*

The following theorem completely resolves the above question.

**Theorem 7.2** *Communication optimal PRMT protocol, tolerating  $\mathcal{A}_{t_b}^{static}$  is possible over a digraph iff there are  $n \geq 2t_b + 1$  wires in the top band and  $u$  wires in the bottom band such that  $(n - 2t_b) + 2u = \Theta(n)$ .*

**PROOF: Necessity:** First of all, irrespective of the value of  $u$ , from Theorem 7.1,  $n \geq 2t_b + 1$ . Hence the digraph must have  $n \geq 2t_b + 1$  wires in the *top band* for the existence of any PRMT that is *communication optimal*. Next we show that  $u$  must satisfy  $(n - 2t_b) + 2u = \Theta(n)$  for the existence of any *communication optimal* PRMT protocol. We have to prove this when  $u < t_b$  because if  $u \geq t_b$  then  $(n - 2t_b) + 2u = \Theta(n)$  is satisfied.

So let  $u < t_b$ . Suppose both **S** and **R** in advance know that the entire bottom band is corrupted. Under this assumption, any multiphase PRMT protocol virtually reduces to a single phase PRMT protocol, where **S** is connected to **R** by  $n \geq 2t_b + 1$  wires, of which at most  $t_b - u$  are corrupted. Now from Theorem 5.5, by substituting  $t_f = 0$ , we find that any single phase protocol must communicate  $\Omega\left(\frac{n\ell}{n-2t_b}\right)$  field elements for reliably sending  $\ell$  field elements, where **S** is connected to **R** by  $n \geq 2t_b + 1$  wires, of which at most  $t_b$  are corrupted. This implies that any single phase protocol must communicate  $\Omega\left(\frac{n\ell}{n-2(t_b-u)}\right)$  field elements for reliably sending  $\ell$  field elements, where **S** is connected to **R** by  $n \geq 2t_b + 1$  wires, of which at most  $t_b - u$  are corrupted. Thus any multiphase PRMT protocol must communicate  $\Omega\left(\frac{n\ell}{n-2(t_b-u)}\right)$  fields elements for reliably sending  $\ell$  field elements over a digraph. Therefore  $\Omega\left(\frac{n\ell}{n-2(t_b-u)}\right)$  defines a lower bound on the communication complexity of any multiphase PRMT protocol sending  $\ell$  field elements. Note that this lower bound is derived by assuming that **S** and **R** in advance know that the entire bottom band is corrupted. Any lower bound derived under this assumption is trivially a lower bound for the more general case, where **S** and **R** does not have this information in advance. Now it is easy to see that  $\Omega\left(\frac{n\ell}{n-2(t_b-u)}\right)$  will turn out to be  $\mathcal{O}(\ell)$  only if  $(n - 2t_b) + 2u = \Theta(n)$ . This completes the necessity proof of Theorem 7.2.

**Sufficiency:** Suppose there exists  $n = 2t_b + 1$  wires in the top band and  $u$  wires in the bottom band, such that  $(n - 2t_b) + 2u = \Theta(n)$ . We then design a three phase PRMT protocol called 3-Optimal-PRMT-Static-Byzantine-Directed, which reliably sends a message  $m^{\mathbf{S}}$  containing  $\ell = \Theta(nt_b)$  field elements by communicating  $\mathcal{O}(nt_b)$  field elements. This will complete the proof of Theorem 7.2. Protocol 3-Optimal-PRMT-Static-Byzantine-Directed will be presented in Section 7.2.2.

### 7.2.1 Black Box Used in Our Protocol

Before presenting our three phase communication optimal PRMT protocol 3-Optimal-PRMT-Static-Byzantine-Directed, we present another single phase protocol called 1-SP-REL-Byzantine, which will be used as a black-box in 3-Optimal-PRMT-Static-Byzantine-Directed. In protocol 1-SP-REL-Byzantine, there are  $n = 2t_b + 1$  wires in the top band and **S** has a message  $m^{\mathbf{S}}$ , which we wants to send to **R**. The protocol has the following properties: If the adversary does at most  $t_b - b$  Byzantine corruptions, then **R** will be able to recover  $m^{\mathbf{S}}$  at the end of the protocol. However, if more than  $t_b - b$  Byzantine faults occur, then **R** will fail to recover  $m^{\mathbf{S}}$  and will detect the presence of more than  $t_b - b$  Byzantine faults.

By closely comparing the steps of protocol 1-SP-REL-Byzantine and protocol 2-SP-REL-Mixed (given in Fig. 5.2), we find that both the protocols are similar, except with the following differences:

1. Protocol 2-SP-REL-Mixed is executed against mixed adversary  $\mathcal{A}_{(t_b, t_f)}^{static}$ , while 1-SP-REL-Byzantine is executed against Byzantine adversary  $\mathcal{A}_{t_b}^{static}$ .
2. In 2-SP-REL-Mixed,  $m^{\mathbf{S}}$  is divided into blocks of size  $k = (N - 2t_b - T) + \frac{t_b}{2} + k_f$ , while in 1-SP-REL-Byzantine,  $m^{\mathbf{S}}$  is divided into blocks of size  $k = (n - 2t_b) + b$ .
3. In 2-SP-REL-Mixed,  $\mathbf{R}$  can reliably send back his findings to  $\mathbf{S}$  (which is either "ERROR1" or "ERROR2" or "SUCCESS" signal) by broadcasting. However, in 1-SP-REL-Byzantine,  $\mathbf{R}$  has no option of reliably sending back his findings to  $\mathbf{S}$ , even by broadcasting, as there may not be sufficient honest wires in the bottom band.

Protocol 1-SP-REL-Byzantine is formally presented in Fig. 7.1.

Figure 7.1: A Single Phase Protocol Based on Error Detection and Error Correction Capability of RS Codes

**Protocol 1-SP-REL-Byzantine** ( $m^{\mathbf{S}}, \ell, n, t_b, b$ ):  $n = 2t_b + 1, 0 \leq b \leq t_b$

1.  $\mathbf{S}$  breaks up  $m^{\mathbf{S}}$  into blocks  $\mathbf{B}_1^{\mathbf{S}}, \dots, \mathbf{B}_z^{\mathbf{S}}$ , each consisting of  $k$  field elements, where  $k = (n - 2t_b) + b$ . If  $\ell$  is not an exact multiple of  $k$ , a default padding is used to make  $\ell \bmod k = 0$ .
2. For  $i = 1, \dots, z$ , corresponding to block  $\mathbf{B}_i^{\mathbf{S}}$ ,  $\mathbf{S}$  computes RS codeword  $C_i^{\mathbf{S}} = (c_{i1}^{\mathbf{S}}, \dots, c_{in}^{\mathbf{S}})$  of length  $n$  and sends  $c_{ij}^{\mathbf{S}}$ , along the wire  $f_j$ , for  $j = 1, \dots, n$ .
3. For  $i = 1, \dots, z$ , let  $\mathbf{R}$  receive  $c_{ij}^{\mathbf{R}}$ , along the wire  $f_j$ , for  $j = 1, \dots, n$ .
4. For  $i = 1, \dots, z$ , let  $C_i^{\mathbf{R}} = (c_{i1}^{\mathbf{R}}, \dots, c_{in}^{\mathbf{R}})$ .  $\mathbf{R}$  executes RS-DEC( $n, C_i^{\mathbf{R}}, t_b - b, b, k$ ).
5. If after correcting  $(t_b - b)$  errors, the RS decoding algorithm does not detect additional errors in any of the  $z$  received vectors, then  $\mathbf{R}$  recovers  $\mathbf{B}_i^{\mathbf{R}} = \mathbf{B}_i^{\mathbf{S}}$ , for  $i = 1, \dots, z$  and concatenates these blocks to recover  $m^{\mathbf{R}} = m^{\mathbf{S}}$ .
6. If  $\exists i \in \{1, 2, \dots, z\}$  such that after correcting  $(t_b - b)$  errors, the decoding algorithm detects additional (at most  $b$ ) errors in the  $i^{\text{th}}$  received vector  $C_i^{\mathbf{R}}$ , then  $\mathbf{R}$  generates "ERROR" signal, which means it has detected that more than  $(t_b - b)$  faults have occurred.

The proof of the properties of protocol 1-SP-REL-Byzantine follows using similar argument as used to prove the properties of protocol 2-SP-REL-Mixed. So to avoid repetition, we do not give the proof and state only the following lemmas:

**Lemma 7.3 (Correctness)** *In protocol 1-SP-REL-Byzantine:*

1. If at most  $(t_b - b)$  Byzantine errors occur in the top band then  $\mathbf{R}$  will be able to recover  $m^{\mathbf{S}}$ .

2. If more than  $(t_b - b)$  Byzantine errors occur in the top band, then  $\mathbf{R}$  will fail to recover  $m^{\mathbf{S}}$ . However, in this case  $\mathbf{R}$  will detect that more than  $(t_b - b)$  Byzantine errors have occurred in the top band.

**Lemma 7.4 (Communication Complexity)** *The communication complexity of protocol 1-SP-REL-Byzantine is  $\mathcal{O}\left(\frac{|m^{\mathbf{S}}|n}{(n-2t_b)+b}\right)$ .*

We are now well equipped to present our communication optimal PRMT protocol 3-Optimal-PRMT-Static-Byzantine, which we do in the next section.

## 7.2.2 A Three Phase Communication Optimal PRMT Protocol

In protocol 3-Optimal-PRMT-Static-Byzantine,  $\mathbf{S}$  sets  $b = \min(\frac{u}{2}, \frac{t_b}{2})$  and executes protocol 1-SP-REL-Byzantine to reliably send  $m^{\mathbf{S}}$ . If  $\mathbf{R}$  is able to recover the message at the end of 1-SP-REL-Byzantine, then  $\mathbf{R}$  sends SUCCESS signal to  $\mathbf{S}$  through the bottom band. On the other hand if  $\mathbf{R}$  fails to recover the message at the end of 1-SP-REL-Byzantine, then it implies that more than  $t_b - b$  Byzantine errors have occurred in the top band. This further implies that the majority of the wires in the bottom band are honest. So  $\mathbf{R}$  sends back the vector in which he has detected the presence of more than  $t_b - b$  Byzantine faults.

Now  $\mathbf{S}$  waits for the feedback from  $\mathbf{R}$  and only considers the feedback, which is received over majority of the wires. But notice that  $\mathbf{S}$  does not know the status of the received feedback. However, the selection of the value of  $b$  in protocol 1-SP-REL-Byzantine allows  $\mathbf{S}$  to take proper response corresponding to the received feedback. The complete formal specification is given in Fig. 7.2.

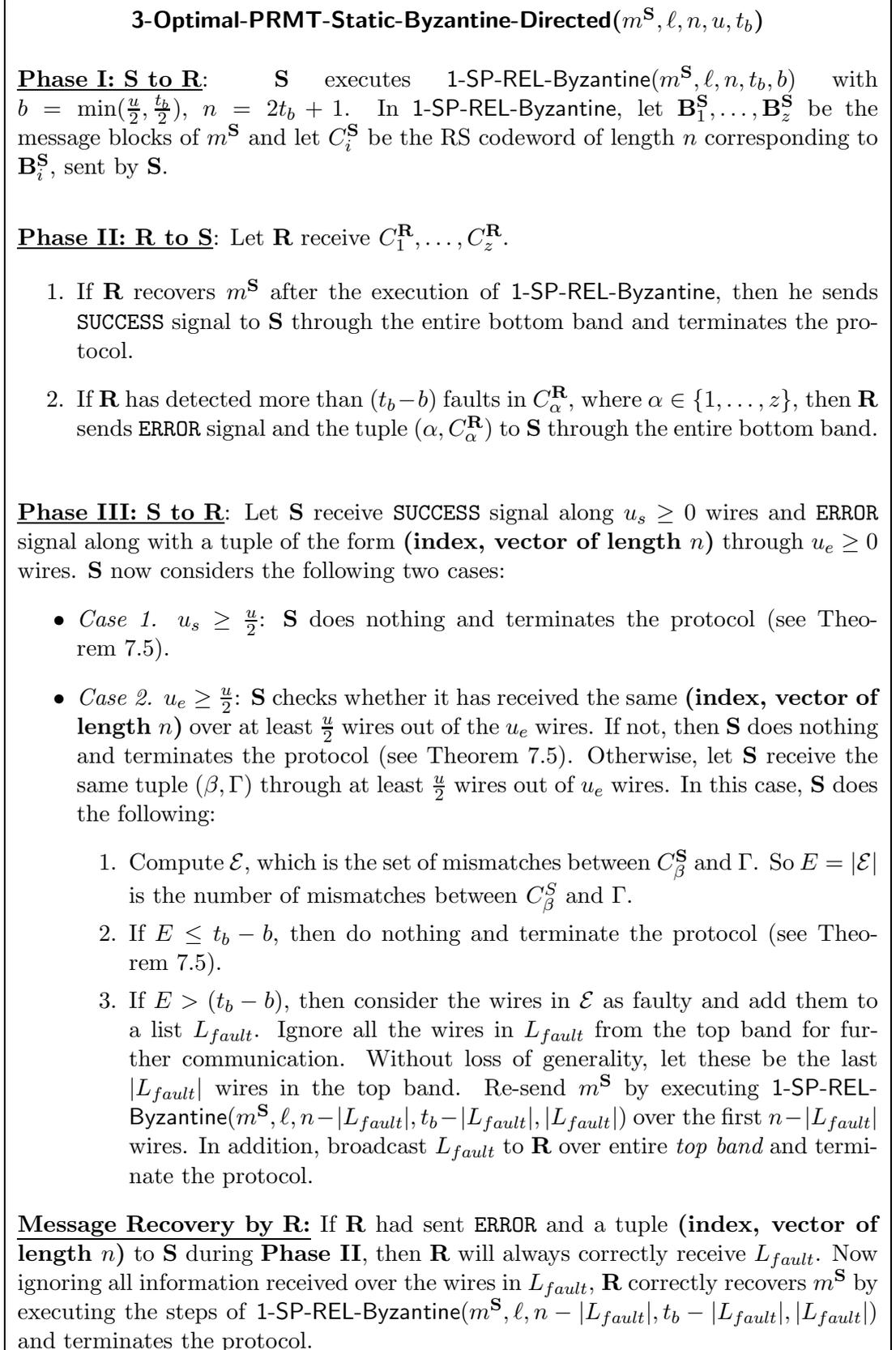
We now prove the properties of protocol 3-Optimal-PRMT-Static-Byzantine-Directed.

**Theorem 7.5 (Correctness)** *Protocol 3-Optimal-PRMT-Static-Byzantine-Directed reliably sends  $m^{\mathbf{S}}$  in at most three phases tolerating  $\mathcal{A}_{t_b}^{static}$ .*

PROOF: We consider the following two cases:

1. **At most  $(t_b - b)$  Byzantine errors took place during Phase I:** In this case, from Lemma 7.3,  $\mathbf{R}$  will be able to recover  $m^{\mathbf{S}}$  correctly at the end of **Phase I**.  $\mathbf{R}$  then sends SUCCESS through the *bottom band* and terminates the protocol. Since it has recovered  $m^{\mathbf{S}}$ , it will simply neglect whatever it receives from  $\mathbf{S}$  during **Phase III**. Hence the theorem holds for this case.
2. **More than  $(t_b - b)$  Byzantine errors took place during Phase I:** In this case, from Lemma 7.3,  $\mathbf{R}$  detects that more than  $(t_b - b)$  Byzantine errors have occurred during **Phase I** and sends ERROR signal along with the tuple  $(\alpha, C_\alpha^{\mathbf{R}})$  through the *bottom band*. Here  $C_\alpha^{\mathbf{R}}$  is the received vector in which  $\mathbf{R}$  has detected more than  $(t_b - b)$  errors. Since more than  $(t_b - b)$  Byzantine errors have occurred in the top band, this implies that in the bottom band, there can be at most  $(b - 1)$  corrupted wires. Since  $b = \min(\frac{t_b}{2}, \frac{u}{2})$ , irrespective of whether  $b = \frac{u}{2}$  or  $\frac{t_b}{2}$ , majority of the wires in the bottom band will be honest. So  $\mathbf{S}$  will correctly receive  $(\alpha, C_\alpha^{\mathbf{R}})$  and ERROR signal over at least  $\frac{u}{2}$  wires. So after locally comparing  $C_\alpha^{\mathbf{R}}$  with  $C_\alpha^{\mathbf{S}}$ ,  $\mathbf{S}$  will come to know the identity of more than  $(t_b - b)$  Byzantine corrupted wires in the top band and adds them to the list  $L_{fault}$ .  $\mathbf{S}$  then broadcasts  $L_{fault}$  to  $\mathbf{R}$  through entire *top band*. Since  $n = 2t_b + 1$ ,  $\mathbf{R}$  correctly receives  $L_{fault}$  and comes to know the identity of more than  $(t_b - b)$  Byzantine corrupted wires in the top band. Finally,  $\mathbf{S}$  re-sends the message by executing 1-SP-REL-Byzantine( $m^{\mathbf{S}}, \ell, n -$

Figure 7.2: A Three Phase Communication Optimal PRMT Protocol in Directed Network Tolerating  $\mathcal{A}_{t_b}^{static}$



$|L_{fault}|, t_b - |L_{fault}|, |L_{fault}|$ ) over the first  $n - |L_{fault}|$  wires<sup>1</sup>. Now since there can be at most  $t_b - |L_{fault}|$  Byzantine corrupted wires in the *top band*, by Lemma 7.3, **R** will be able to recover  $m^S$  at the end of **Phase III**. Hence the theorem holds for this case also.  $\square$

**Theorem 7.6 (Communication Complexity)** *Protocol 3-Optimal-PRMT-Static-Byzantine-Directed is a communication optimal PRMT protocol which reliably sends a message containing  $\ell = \Theta(nt_b)$  field elements by communicating  $\mathcal{O}(nt_b)$  field elements tolerating  $\mathcal{A}_{t_b}^{static}$ .*

PROOF: Since  $n = 2t_b + 1, \ell = |m^S| = \Theta(nt_b), n - 2t_b + 2u = \Theta(n)$  and  $b = \min(\frac{u}{2}, \frac{t_b}{2})$ , from Lemma 7.4, the communication complexity of **Phase I** is  $\mathcal{O}(nt_b)$ . During **Phase II**, **R** either sends SUCCESS signal or a tuple (**index, vector of length  $n$** ), along with ERROR signal over all the  $u$  wires in bottom band. This involves communication of at most  $nu = \mathcal{O}(nt_b)$  field elements. During **Phase III**, **S** either sends nothing or re-sends the message. Communication complexity of re-sending the message is  $\mathcal{O}(\frac{(n-|L_{fault}|)|m^S|}{|L_{fault}|})$ . Since  $|L_{fault}| > (t_b - b)$ , irrespective of whether  $b = \frac{u}{2}$  or  $b = \frac{t_b}{2}$ , the following holds:  $|L_{fault}| = \Theta(t_b)$  and  $n - |L_{fault}| = \Theta(t_b)$ . Hence re-sending  $m^S$  incurs a communication complexity of  $\mathcal{O}(nt_b)$ . Thus the total communication complexity of 3-Optimal-PRMT-Static-Byzantine-Directed is  $\mathcal{O}(nt_b)$ .  $\square$

### 7.3 Concluding Remarks and Open Problems

In this chapter, we have studied PRMT in directed network tolerating  $\mathcal{A}_{t_b}^{static}$ . Specifically, we have characterized the class of digraphs over which PRMT protocol with constant factor overhead is possible. However, the characterization holds only for static Byzantine adversary. This brings forth the following open problems:

**Open Problem 6** *Let  $\mathcal{N}$  be a directed network where there are  $n$  wires in the top band and  $u$  wires in the bottom band, such that at most  $t_b$  and  $t_f$  wires are under the control of a threshold static adversary  $\mathcal{A}_{(t_b, t_f)}^{static}$  in Byzantine and failstop fashion respectively. Then what are the necessary and sufficient conditions that  $n$  and  $u$  should satisfy for the existence of any PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ , which sends a message containing  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements?*

**Open Problem 7** *Let  $\mathcal{N}$  be a directed network where there are  $n$  wires in the top band and  $u$  wires in the bottom band, such that at most  $t_b$  and  $t_f$  wires are under the control of a threshold mobile adversary  $\mathcal{A}_{(t_b, t_f)}^{mobile}$  in Byzantine and failstop fashion respectively. Then what are the necessary and sufficient conditions that  $n$  and  $u$  should satisfy for the existence of any PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ , which sends a message containing  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements?*

Till now, we have focussed on PRMT where it is required that **R** should output the message without any error. In the next chapter, we discuss about SRMT, where **R** is allowed to output an incorrect message with a negligible error probability.

---

<sup>1</sup>Recall that in the protocol, we have assumed that the last  $|L_{fault}|$  wires are Byzantine corrupted. This is without loss of generality.

## Chapter 8

# SRMT in Undirected Networks Tolerating Static Mixed Adversary

In PRMT, it is required that  $\mathbf{R}$  should correctly output the message at the end of the protocol without any error. On the other hand, in SRMT a negligible error probability is allowed in the outcome of the protocol. It is a well-known fact that in several problem domains, allowing a negligible error probability in the outcome helps to a great extent in arriving at more efficient and simpler solutions than their deterministic counterpart. The problem domains range from famous number theoretic randomised primality testing algorithms [65] to various distributed computation tasks like verifiable secret sharing (VSS) [66, 8, 23, 52], multiparty computation (MPC) [66, 8, 9, 56] to name a few. Motivated by this, Franklin et al. studied PRMT with negligible error probability in [33]. The issue of POSSIBILITY, FEASIBILITY and OPTIMALITY of SRMT in undirected synchronous network tolerating threshold static mixed adversary was completely resolved in [75]. For the sake of completeness, we recall these results and present them in the next section. The main purpose of recalling these results is to make the thesis self contained. This is because several of these results and techniques will be later used in other chapters of our thesis.

We now give the formal specification of the network settings used in this chapter.

### 8.1 Network Model and Adversary Settings

The network model used in this chapter is similar to the one used in Chapter 5. Thus there are  $n$  bidirectional synchronous wires  $w_1, \dots, w_n$  between  $\mathbf{S}$  and  $\mathbf{R}$ , of which at most  $t_b$  and  $t_f$  wires can be under the control of a static mixed adversary  $\mathcal{A}_{(t_b, t_f)}^{static}$  in Byzantine and fail-stop fashion respectively. We assume that all computation and communication are done over a finite field  $\mathbb{F}$ , where  $\mathbb{F} = GF(2^\kappa)$ . Here  $\kappa$  is the error parameter<sup>1</sup>. Thus each field element can be represented by  $\mathcal{O}(\kappa)$  bits. Moreover, without loss of generality, we assume that  $n = \text{poly}(\kappa)$ .

#### 8.1.1 Tools Used in SRMT and SSMT Protocols

In all the SRMT and SSMT protocols presented in this thesis, we will be using the following tools:

---

<sup>1</sup>Recall that in SRMT,  $\mathbf{R}$  should output the correct message, except with error probability  $2^{-\Omega(\kappa)}$ .

**Definition 8.1 (Unconditionally Reliable Authentication [66, 24, 33])** *It is used to send a message  $M$  over a wire such that if the wire is uncorrupted, then the receiver correctly gets  $M$ . On the other hand, if the wire is corrupted, then the receiver does not get  $M$  but is able to detect the corruption with very high probability. Though there are several implementations for this well known primitive, we use the following implementation in this thesis: Let a random, non-zero  $(a, b) \in \mathbb{F}^2$  be securely established between the sender and the receiver in advance. The sender computes  $x = URauth(M; a, b) = aM + b$  and sends  $(M, x)$  to the receiver over the wire. Let the receiver receive  $(M', x')$  along the wire. Receiver verifies  $x' \stackrel{?}{=} URauth(M'; a, b)$ . If the test fails then the receiver concludes that  $M' \neq M$ , otherwise  $M' = M$ . The tuple  $(a, b)$  is called authentication key. The probability that  $M' \neq M$ , but still the receiver fails to detect it is at most  $\frac{1}{|\mathbb{F}|}$ , which is negligible in our context. Note that the key  $a$  remains information theoretically secure, even if the adversary knows  $(M, x)$  by eavesdropping the wire.*

**Definition 8.2 (Unconditionally Secure Authentication [66, 24, 33])** *The goal here is similar to unconditionally reliable authentication. However, we now require an additional requirement that  $M$  should be information theoretically secure, even if the wire is under the control of the adversary. Again there are several implementations for this well known primitive. In this thesis, we use the following implementation: Let a random, non-zero  $(a, b, c) \in \mathbb{F}^3 - \{(0, 0, 0)\}$  be securely established between the sender and the receiver in advance. The sender computes  $(x, y) = USauth(M; a, b, c) = (M + a, b(M + a) + c)$  and sends  $(x, y)$  to the receiver over the wire. Let the receiver receive  $(x', y')$  along the wire. The receiver verifies  $y' \stackrel{?}{=} bx' + c$ . If the test fails then the receiver concludes that the wire is corrupted, else the receiver recovers  $x' - a$ . It is easy to see that even if the adversary knows  $(x, y)$ , then also  $M$  is information theoretically secure. Moreover, if  $(x', y') \neq (x, y)$ , then except with error probability  $\frac{1}{|\mathbb{F}|}$  (which is negligible), the receiver will be able to detect it.*

**Definition 8.3 (Unconditional Hashing [9])** *Let  $(v_1, v_2, \dots, v_\ell)$  be a random vector from  $\mathbb{F}^\ell$ , where  $\ell > 1$  and  $k \in \mathbb{F} - \{0\}$ . Then we define  $hash(k; v_1, v_2, \dots, v_\ell) = v_1 + v_2k + v_3k^2 + \dots + v_\ell k^{\ell-1}$ . Here  $k$  is called the hash key. The probability that two different vectors map to the same hash value for a uniformly chosen hash key is at most  $\frac{\ell}{|\mathbb{F}|} \approx 2^{-\Omega(\kappa)}$ . If the adversary knows only  $k$  and  $hash(k; v_1, v_2, \dots, v_\ell)$ , then  $\ell - 1$  elements in the vector will be information theoretically secure.*

## 8.2 Characterization of SRMT and Bounds on the Communication Complexity of SRMT

We now recall the existing results regarding the characterization of SRMT.

### 8.2.1 Characterization of SRMT in Undirected Networks

As stated earlier, SRMT problem was first defined in [33], where the authors studied SRMT in the presence of a static Byzantine adversary. Specifically, the authors have shown the following:

**Theorem 8.4 ([33])** *SRMT in an undirected synchronous network tolerating a threshold static Byzantine adversary  $\mathcal{A}_{t_b}^{static}$  is possible iff there exists  $n \geq 2t_b + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ .*

In [75], the author extended the above characterization for static mixed adversary and showed the following:

**Theorem 8.5 ([75])** *SRMT in an undirected synchronous network tolerating a threshold static mixed adversary  $\mathcal{A}_{(t_b, t_f)}^{static}$  is possible iff there exists  $n \geq 2t_b + t_f + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ .*

**Comparison 8.6 (Connectivity Requirement of PRMT and SRMT)** *Comparing Theorem 5.1 and Theorem 8.5, we find that the connectivity requirement of PRMT as well as SRMT is same. That is, allowing a negligible error probability does not reduce the connectivity requirement for reliable message transmission.*

Though allowing a negligible error probability does not reduce the connectivity requirement, it does reduce the communication complexity of RMT protocols. We first consider the case of single phase protocols.

### 8.2.2 Lower Bound and Upper Bound for Single Phase SRMT

In [75], the author proved the lower bound on the communication complexity of single phase SRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ , which is given by the following theorem:

**Theorem 8.7 ([75])** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + t_f + 1$  wires. Then any single phase SRMT protocol must communicate  $\Omega\left(\frac{n\ell}{n-(t_b+t_f)}\right)$  field elements to transmit a message containing  $\ell$  field elements tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ . In terms of bits, any single phase SRMT protocol must communicate  $\Omega\left(\frac{n\ell\kappa}{n-(t_b+t_f)}\right)$  bits to reliably send a message containing  $\ell\kappa$  bits.*

**Comparison 8.8 (Communication Complexity of PRMT and SRMT)** *While the lower bound on the communication complexity of any single phase PRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$  is  $\Omega\left(\frac{n\ell}{n-(2t_b+t_f)}\right)$  field elements (see Theorem 5.5), the same for SRMT is  $\Omega\left(\frac{n\ell}{n-(t_b+t_f)}\right)$  field elements (Theorem 8.7). Recall that as pointed out in Comparison 8.6, the connectivity requirement for both PRMT and SRMT tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$  is  $n \geq 2t_b + t_f + 1$ . Assuming  $n = 2t_b + t_f + 1$ , the lower bound for single phase PRMT and SRMT become  $\Omega(n\ell)$  and  $\Omega\left(\frac{n\ell}{t_b}\right)$  field elements respectively. Now if  $t_b = \Theta(n)$  then the lower bound for single phase SRMT becomes  $\Omega(\ell)$  field elements. This implies that for  $t_b = \Theta(n)$ , communication of  $\ell$  field elements requires transmission of  $\Omega(n\ell)$  field elements for PRMT and  $\Omega(\ell)$  field elements for SRMT. Now notice that PRMT and SRMT tolerating Byzantine adversary  $\mathcal{A}_{t_b}^{static}$  (i.e.,  $t_f = 0$ ) requires  $n \geq 2t_b + 1$  wires. If  $n = 2t_b + 1$ , then  $t_b = \Theta(n)$  holds. Hence the conclusion is that in the presence of  $\mathcal{A}_{t_b}^{static}$  the lower bound on the communication complexity of any single phase PRMT and SRMT are  $\Omega(n\ell)$  and  $\Omega(\ell)$  field elements respectively. This clearly shows that allowing a negligible error probability helps in significant reduction in the lower bound on the communication complexity of RMT protocols.*

The author in [75] showed that the bound on the communication complexity as given in Theorem 8.7 is asymptotically tight. Specifically, the author presented a communication optimal single phase SRMT protocol over  $n = 2t_b + t_f + 1$  wires, which we call as 1-Optimal-SRMT-Static-Mixed. The protocol delivers a message containing  $(t_b + 1)n$  field elements by communicating  $\mathcal{O}(n^2)$  field elements. Before presenting the

protocol, we will present a technique called **Extrapolation Technique**, which is used protocol 1-Optimal-SRMT-Static-Mixed. The same technique will be again used in several other SRMT and SSMT protocols in subsequent chapters.

### 8.2.2.1 Extrapolation Technique

The settings used in the Extrapolation Technique are as follows:

1.  $\mathbf{S}$  has a block  $B_{init}$ , consisting of  $ROW \times COL$  random elements from  $\mathbb{F}$ . Here  $ROW$  and  $COL$  are variables.
2. The elements of  $B_{init}$  are assumed to be arranged in the form of a  $ROW \times COL$  matrix, where for  $i = 1, \dots, ROW$ , the  $i^{th}$  row of  $B_{init}$  is  $(r_{i,1}^{\mathbf{S}}, \dots, r_{i,COL}^{\mathbf{S}})$ .

Now using the Extrapolation Technique,  $\mathbf{S}$  constructs an  $N \times COL$  matrix  $B_{ext}$  (where  $N > ROW$ ) by executing the steps given in Fig. 8.1.

Figure 8.1: Steps for the Extrapolation Technique

**Extrapolation Technique**( $ROW, COL, N, B_{init}$ )

1. For  $j = 1, \dots, COL$ ,  $\mathbf{S}$  constructs a polynomial  $q_j^{\mathbf{S}}(x)$  of degree  $(ROW - 1)$ , passing through the elements of  $j^{th}$  column of  $B_{init}$ . That is,  $q_j^{\mathbf{S}}(x)$  passes through the points  $(1, r_{1,j}^{\mathbf{S}}), \dots, (ROW, r_{ROW,j}^{\mathbf{S}})$ .
2. For  $j = 1, \dots, COL$ ,  $\mathbf{S}$  evaluates the polynomial  $q_j^{\mathbf{S}}(x)$  at  $x = ROW + 1, \dots, N$  to get  $c_{ROW+1,j}^{\mathbf{S}}, \dots, c_{N,j}^{\mathbf{S}}$  respectively.
3. Now  $B_{ext}$  is the  $N \times COL$  matrix, whose  $j^{th}$  column is  $(r_{1,j}^{\mathbf{S}}, \dots, r_{ROW,j}^{\mathbf{S}}, c_{ROW+1,j}^{\mathbf{S}}, \dots, c_{N,j}^{\mathbf{S}})^T$ .

Pictorially, the matrix  $B_{ext}$ , as constructed from  $B_{init}$  using Extrapolation Technique, is shown in Fig. 8.2.

Figure 8.2: The  $N \times COL$  Matrix  $B_{ext}$  as Constructed from the  $ROW \times COL$  Matrix  $B_{init}$

$r_{1,1}^{\mathbf{S}}$	$\dots$	$r_{1,j}^{\mathbf{S}}$	$\dots$	$r_{1,COL}^{\mathbf{S}}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$r_{i,1}^{\mathbf{S}}$	$\dots$	$r_{i,j}^{\mathbf{S}}$	$\dots$	$r_{i,COL}^{\mathbf{S}}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$r_{ROW,1}^{\mathbf{S}}$	$\dots$	$r_{ROW,j}^{\mathbf{S}}$	$\dots$	$r_{ROW,COL}^{\mathbf{S}}$
$c_{ROW+1,1}^{\mathbf{S}}$	$\dots$	$c_{ROW+1,j}^{\mathbf{S}}$	$\dots$	$c_{ROW+1,COL}^{\mathbf{S}}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$c_{N,1}^{\mathbf{S}}$	$\dots$	$c_{N,j}^{\mathbf{S}}$	$\dots$	$c_{N,COL}^{\mathbf{S}}$

The following results taken from [75] states the properties of the Extrapolation Technique.

**Lemma 8.9** *In  $B_{ext}$ , all the  $N$  elements of any column can be uniquely generated from any  $ROW$  elements of the same column.*

PROOF: The proof follows from the simple observation that the  $N$  elements along any column of  $B_{ext}$  lie on a  $(ROW - 1)$  degree polynomial and any  $ROW$  points on a  $(ROW - 1)$  degree polynomial are enough to reconstruct the  $ROW$  degree polynomial.  $\square$

**Lemma 8.10** *The elements of  $B_{init}$  can be uniquely determined from any  $ROW$  rows of  $B_{ext}$ .*

PROOF: From the construction of  $B_{ext}$ , the elements of  $B_{init}$  are arranged in the first  $ROW$  rows. If the first  $ROW$  rows are known then the lemma holds trivially. On the other hand, if some other  $ROW$  rows are known, then from Lemma 8.9, the  $j^{th}$  column,  $1 \leq j \leq COL$ , of  $B_{ext}$  can be completely generated from  $ROW$  elements of the same column. Hence, knowledge of any  $ROW$  rows can reconstruct the whole matrix  $B_{ext}$  and hence the matrix  $B_{init}$  (which is just the first  $ROW$  rows of  $B_{ext}$ ).  $\square$

We are now well equipped to present protocol 1-Optimal-SRMT-Static-Mixed, which we do in the next section.

### 8.2.2.2 Protocol 1-Optimal-SRMT-Static-Mixed: Single Phase Communication Optimal SRMT Protocol Tolerating $\mathcal{A}_{(t_b, t_f)}^{static}$

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + t_f + 1$  wires, denoted by  $w_1, \dots, w_n$ . Let  $\mathbf{S}$  has a message  $m^{\mathbf{S}}$  containing  $(t_b + 1) \times n$  elements from  $\mathbb{F}$ . We then present protocol 1-Optimal-SRMT-Static-Mixed, which allows  $\mathbf{S}$  to reliably send  $m^{\mathbf{S}}$  with very high probability in a single phase by communicating  $\mathcal{O}(n^2)$  field elements. The protocol is given in Fig. 8.3.

We now prove the properties of protocol 1-Optimal-SRMT-Static-Mixed.

**Lemma 8.11** *In protocol 1-Optimal-SRMT-Static-Mixed if  $\mathbf{R}$  concludes that  $F_i^{\mathbf{R}}$  is a valid row of  $B_{ext}$ , then except with error probability  $2^{-\Omega(\kappa)}$ ,  $F_i^{\mathbf{R}} = F_i^{\mathbf{S}}$ .*

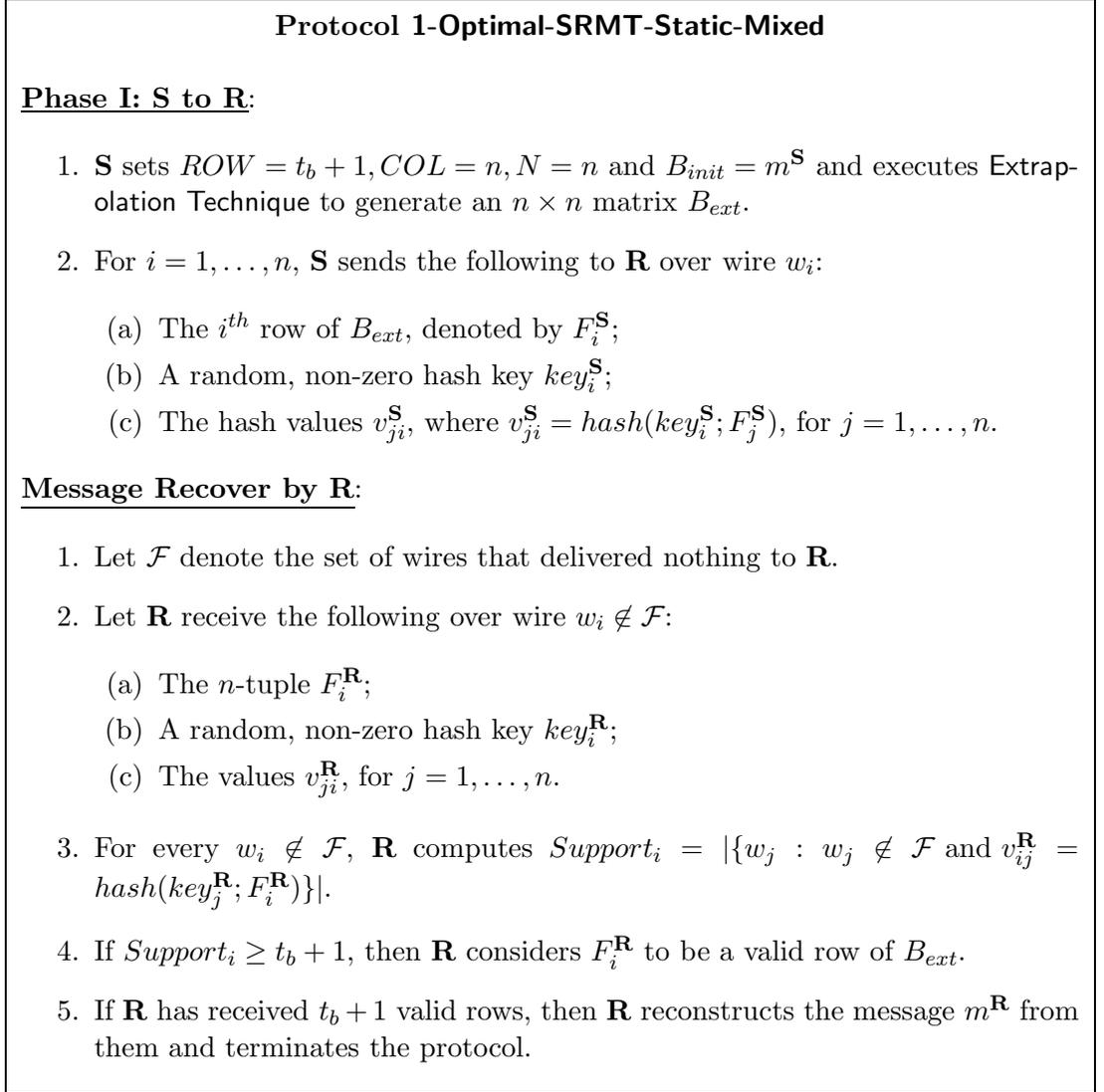
PROOF: The lemma is true without any error if wire  $w_i$  is uncorrupted. So let wire  $w_i$  be a corrupted wire, who delivers  $F_i^{\mathbf{R}} \neq F_i^{\mathbf{S}}$ . In this case, if  $F_i^{\mathbf{R}}$  is considered as a valid row of  $B_{ext}$ , then it implies that  $Support_i \geq t_b + 1$ . Since there can be at most  $t_b$  Byzantine corrupted wires, this implies that there exists at least one honest wire, say  $w_j$ , which correctly and securely delivered the hash key  $key_j^{\mathbf{R}} = key_j^{\mathbf{S}}$  and hash value  $v_{ij}^{\mathbf{R}} = v_{ij}^{\mathbf{S}} = hash(key_j^{\mathbf{S}}; F_i^{\mathbf{S}}) = hash(key_j^{\mathbf{R}}; F_i^{\mathbf{S}})$ , such that  $w_j \in Support_i$ . Since  $w_j \in Support_i$ , it implies that  $v_{ij}^{\mathbf{R}} = hash(key_j^{\mathbf{R}}; F_i^{\mathbf{R}})$ . Since adversary does not know  $key_j^{\mathbf{R}}$  and  $v_{ij}^{\mathbf{R}}$ , he can ensure that  $v_{ij}^{\mathbf{R}} = hash(key_j^{\mathbf{R}}; F_i^{\mathbf{S}})$ , as well as  $v_{ij}^{\mathbf{R}} = hash(key_j^{\mathbf{R}}; F_i^{\mathbf{R}})$ , where  $F_i^{\mathbf{R}} \neq F_i^{\mathbf{S}}$ , with probability at most  $\frac{n-1}{|\mathbb{F}|} \approx 2^{-\Omega(\kappa)}$ , which is negligible in our context. So with very high probability,  $w_j$  will not belong to  $Support_i$ , which is a contradiction. So with overwhelming probability  $F_i^{\mathbf{R}} = F_i^{\mathbf{S}}$ .  $\square$

**Lemma 8.12** *In protocol 1-Optimal-SRMT-Static-Mixed, if  $\mathbf{R}$  gets  $t_b + 1$  valid rows of  $B_{ext}$  then  $\mathbf{R}$  can recover  $m^{\mathbf{S}}$ .*

PROOF: The proof follows from Lemma 8.10 and the fact that  $ROW = t_b + 1$  in protocol 1-Optimal-SRMT-Static-Mixed.  $\square$

**Lemma 8.13** *In protocol 1-Optimal-SRMT-Static-Mixed, except with error probability  $2^{-\Omega(\kappa)}$ ,  $m^{\mathbf{R}} = m^{\mathbf{S}}$ .*

Figure 8.3: Single Phase Protocol 1-Optimal-SRMT-Static-Mixed Tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$



PROOF: First of all notice that **R** will always output some message. This is because there always exist  $n - (t_b + t_f) = t_b + 1$  honest wires, which will always deliver valid rows of  $B_{ext}$ . Moreover, even if a wire which is under the control of the adversary has delivered a valid row, then from Lemma 8.11, the row is indeed a valid row of  $B_{ext}$ , except with probability  $2^{-\Omega(\kappa)}$ . This implies that with very high probability, the  $t_b + 1$  valid rows used by **R** to recover  $m^{\mathbf{R}}$  are indeed the rows of  $B_{ext}$ . Thus, except with error probability  $2^{-\Omega(\kappa)}$ ,  $m^{\mathbf{R}} = m^{\mathbf{S}}$ .  $\square$

**Lemma 8.14** *In protocol 1-Optimal-SRMT-Static-Mixed, **S** communicates  $\mathcal{O}(n^2\kappa)$  bits.*

PROOF: Over each wire, **S** sends a row of  $B_{ext}$  consisting of  $n$  elements, a hash key and  $n$  hash values. So overall, **S** sends  $\mathcal{O}(n^2)$  field elements to **R**. Since each field element can be represented by  $\mathcal{O}(\kappa)$  bits, **S** communicates  $\mathcal{O}(n^2\kappa)$  bits.

**Theorem 8.15** *Protocol 1-Optimal-SRMT-Static-Mixed is a communication optimal SRMT protocol, which reliably sends a message containing  $\Theta(nt_b\kappa)$  bits by communicating  $\mathcal{O}(n^2\kappa)$  bits, tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ .*

PROOF: The proof that 1-Optimal-SRMT-Static-Mixed is an SRMT protocol follows from Lemma 8.11, Lemma 8.12 and Lemma 8.13. If  $n = 2t_b + t_f + 1$ , then from Theorem 8.7, any single phase SRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$  must communicate  $\Omega(n^2\kappa)$  bits to reliably send  $\ell = \Theta(nt_b\kappa)$  bits. Since the communication complexity of protocol 1-Optimal-SRMT-Static-Mixed is  $\mathcal{O}(n^2\kappa)$  bits, protocol 1-Optimal-SRMT-Static-Mixed is a communication optimal SRMT protocol.  $\square$

### 8.2.3 Lower Bound and Upper Bound for Multi Phase SRMT

If more than one phase is allowed, then the communication complexity of SRMT protocols can be reduced. The lower bound on the communication complexity of multiphase SRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$  is given by the following theorem:

**Theorem 8.16** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + t_f + 1$  wires. Then any multi phase SRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$  must communicate  $\Omega(\ell)$  field elements to transmit a message containing  $\ell$  field elements tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ . In terms of bits, any multi phase SRMT protocol must communicate  $\Omega(\ell\kappa)$  bits to reliably send a message containing  $\ell\kappa$  bits.*

PROOF: Easy because any SRMT protocol has to at least send the message.  $\square$

The bound given in the above theorem is asymptotically tight, as shown in the following theorem:

**Theorem 8.17** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + t_f + 1$  wires. Then there exists an  $\mathcal{O}(\log \frac{t_f}{t_b})$  phase SRMT protocol, tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ , which reliably sends a message containing  $\ell = \Theta(n^3)$  field elements by communicating  $\mathcal{O}(n^3)$  field elements.*

PROOF: The proof follows from Corollary 5.17.2 and the fact that any PRMT protocol is also an SRMT protocol.  $\square$

## 8.3 Concluding Remarks and Open Problems

In this chapter, we have discussed about the POSSIBILITY, FEASIBILITY and OPTIMALITY of SRMT in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ . The results are summarized in Fig. 8.4. The conclusion drawn from these results is the following: *Allowing negligible error probability in the reliability does not reduce the connectivity requirement for reliable message transmission. However it does reduce the communication complexity of RMT protocols.*

From Fig. 8.4, we find that protocol 1-Optimal-SRMT-Static-Mixed is communication optimal only if  $\ell = \Theta(nt_b)$ . This brings forth the following open question:

**Open Problem 8** *Let there exists  $n \geq 2t_b + t_f + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$  under the influence of  $\mathcal{A}_{(t_b, t_f)}^{static}$ . Then does there exist a single phase communication optimal SRMT protocol, which reliably sends a message containing  $\ell$  field elements ( $\ell$  can be anything) by communicating  $\mathcal{O}\left(\frac{n\ell}{n-(t_b+t_f)}\right)$  field elements?*

To show the tightness of the bound given in Theorem 8.16, we actually showed the presence of a PRMT protocol, which takes  $\mathcal{O}(\log \frac{t_f}{t_b})$  phases. This leads to the second open problem of the chapter, which is as follows:

Figure 8.4: Summary of the Results for SRMT in Undirected Synchronous Network Tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$

Number of Phases ( $r$ )	Connectivity Requirement ( $n$ )	Lower Bound on Communication Complexity	Upper Bound
$r = 1$	$n \geq 2t_b + t_f + 1$ Theorem 8.5	$\Omega\left(\frac{n\ell}{n-(t_b+t_f)}\right)$ Theorem 8.7	Protocol 1-Optimal-SRMT-Static-Mixed: $n = 2t_b + t_f + 1, \ell = \Theta(nt_b)$ Communication complexity = $\mathcal{O}\left(\frac{n\ell}{t_b}\right)$
$r \geq 2$	$n \geq 2t_b + t_f + 1$ Theorem 8.5	$\Omega(\ell)$ Theorem 8.16	Theorem 8.17

**Open Problem 9** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + t_f + 1$  wires. Then does there exist an SRMT protocol tolerating  $\mathcal{A}_{(t_b, t_f)}^{static}$ , which reliably sends a message containing  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements and takes less than  $\mathcal{O}(\log \frac{t_f}{t_b})$  phases?*

We do not discuss about SRMT in directed network at this point. This is because our SRMT protocol in directed network will use some SSMT protocols in directed networks as black-box. So for better understanding, we will discuss about SRMT and SSMT in directed networks later in the same chapter.

With this, we come to the end of first part of our thesis. Till now, we have considered only RMT protocols, where there is no issue of security. The next part deals with SMT protocols, where we require both reliability as well as security.

## Part II

# Results for PSMT and SSMT in Synchronous Network

## Chapter 9

# PSMT in Undirected Networks Against Static Byzantine Adversary

In this chapter, we focus on PSMT in *undirected synchronous network, tolerating threshold static Byzantine adversary*, denoted by  $\mathcal{A}_{t_b}^{static}$ . The PSMT problem in this model has been studied by several researchers [28, 70, 77, 4, 58, 30, 42]. Recall that in PSMT, in addition to perfect reliability, we also require perfect security. The contribution of this chapter is two fold. First, we present all relevant known bounds, thus making the available literature more accessible. As a second contribution, we present a three phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ . Till the publication of this result in [58], our three phase PSMT protocol was the only efficient communication optimal PSMT in undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{static}$ . However, recently in [42], Kurosawa et al. have improved the phase complexity of our protocol by reducing the number of phases by one. Nevertheless, we still present our three phase protocol to highlight some techniques, which will be later used in other PSMT protocols of the thesis.

The network model and adversary settings used in this chapter are exactly same as in Chapter 3. Thus, we assume that  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n$  bi-directional synchronous wires  $w_1, \dots, w_n$ , of which at most  $t_b$  wires can be under the control of  $\mathcal{A}_{t_b}^{static}$ . We now present the existing results for PSMT in undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{static}$ .

### 9.1 Existing Results for PSMT Tolerating $\mathcal{A}_{t_b}^{static}$

We begin with the description of the existing results for single phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$ .

#### 9.1.1 Single Phase PSMT Tolerating $\mathcal{A}_{t_b}^{static}$

The PSMT problem was first introduced in [28]. Dolev et al. [28] gave the following necessary and sufficient condition for the existence of single phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$ .

**Theorem 9.1 ([28])** *Any single phase PSMT in undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff there exists  $n \geq 3t_b + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ .*

PROOF (SKETCH): The necessity is proved by showing that in any single phase PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , the information sent by  $\mathbf{S}$  over any  $n - 2t_b$  wires should have full information about the message. Since in any single phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$  the adversary can passively listen at most  $t_b$  wires, the above condition implies that in any single phase PSMT,  $n - 2t_b > t_b$  should hold, otherwise adversary will also know the secret message. This further implies that  $n > 3t_b$  should hold.

The sufficiency is shown as follows: suppose there exists  $n = 3t_b + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ . To securely send a message  $m^{\mathbf{S}}$ , sender  $\mathbf{S}$  selects a random polynomial of degree  $t_b$  whose constant term is  $m^{\mathbf{S}}$  and computes an RS codeword of length  $n$  from this polynomial.  $\mathbf{S}$  then sends one component of the codeword to  $\mathbf{R}$  over each wire.  $\mathbf{R}$  will receive  $3t_b + 1$  points on a polynomial of degree  $t_b$  and at most  $t_b$  points could be corrupted. However, by applying RS-DEC,  $\mathbf{R}$  will be able to correct these errors and recover the original polynomial and hence the message  $m^{\mathbf{S}}$ . The secrecy of  $m^{\mathbf{S}}$  follows from the fact that  $\mathcal{A}_{t_b}^{static}$  will know only  $t_b$  points on the polynomial, whose degree is  $t_b$ . Thus adversary will lack by one point to uniquely interpolate the polynomial, implying information theoretic security for  $m^{\mathbf{S}}$ .  $\square$

The above theorem resolves the issue of POSSIBILITY of single phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$ . The lower bound on the communication complexity of single phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$  was proved in two independent works in [81, 30].

**Theorem 9.2 ([81, 30])** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 3t_b + 1$  wires. Then any single phase PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  must communicate  $\Omega\left(\frac{n\ell}{n-3t_b}\right)$  field elements to securely send a message containing  $\ell$  field elements. Moreover, this bound is tight.*

PROOF (SKETCH): The lower bound is derived by first showing that the communication complexity of any single phase PSMT to securely send a message  $m^{\mathbf{S}}$  tolerating  $\mathcal{A}_{t_b}^{static}$  is not less than the share complexity (sum of the length of the shares) of a secret sharing scheme, which generates  $n$  shares for  $m^{\mathbf{S}}$ , such that any set of  $n - 2t_b$  shares has full information about the message, while any set of  $t_b$  shares has no information about the message. Then it is shown that the share complexity (sum of the length of the shares) of any secret sharing scheme, which generates  $n$  shares for  $m^{\mathbf{S}}$  containing  $\ell$  field elements, such that any set of  $n - 2t_b$  shares has full information about the message, while any set of  $t_b$  shares has no information about the message is  $\Omega\left(\frac{n\ell}{n-3t_b}\right)$ . This is done by using similar arguments as given in Lemma 6.11.

The tightness of the bound follows from the protocol which is sketched in Theorem 9.1, which securely sends a message containing  $\ell = 1$  field element by communicating  $\mathcal{O}(n)$  field elements, where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 3t_b + 1$  wires.  $\square$

### 9.1.2 Multi Phase PSMT Tolerating $\mathcal{A}_{t_b}^{static}$

The issue of POSSIBILITY for multi phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$  was resolved in [28], where the authors gave the following characterization:

**Theorem 9.3 ([28])** *Let  $r \geq 2$ . Then any  $r$ -phase PSMT protocol in undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff there exists  $n \geq 2t_b + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ .*

PROOF (SKETCH): The necessity is proved by showing that in any multiphase phase PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , the information sent by  $\mathbf{S}$  over any  $n - t_b$  wires should

have full information about the message. Since in any PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  the adversary can passively listen at most  $t_b$  wires, the above condition implies that in any multi phase PSMT protocol,  $n - t_b > t_b$  should hold, otherwise adversary will also know the secret message. This further implies that  $n > 2t_b$  should hold.

To show the sufficiency, Dolev et al. presented three PSMT protocols in [28] with  $2t_b + 1$  bidirectional wires between  $\mathbf{S}$  and  $\mathbf{R}$ .

1. The first protocol requires  $t_b + 1$  phases and communicates  $\mathcal{O}(n^3\ell)$  field elements to securely send  $\ell$  field elements;
2. The second protocol takes three phases and communicates  $\mathcal{O}(n^5\ell)$  field elements to securely send  $\ell$  field elements;
3. The last protocol takes two phases and communicates  $\mathcal{O}(n^3\ell)$  field elements to securely send  $\ell$  field elements. Unfortunately, in this protocol,  $\mathbf{S}$  and  $\mathbf{R}$  has to perform *exponential* computation.  $\square$

After the work of Dolev et al., several attempts were made to improve the communication complexity of multi phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$ . The first improvement was achieved by Sayeed et al. in [70], who presented a *two* phase, *polynomial* time PSMT protocol in undirected synchronous network, where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + 1$  wires. The protocol of [70] requires a communication complexity of  $\mathcal{O}(n^3\ell)$  field elements to securely send  $\ell$  field elements. This significantly improved the inefficient (third) protocol of [28]. In another work, Srinathan et al. [85] improved the first protocol of [28] by designing a PSMT protocol with  $n = 2t_b + 1$  wires, which takes  $\mathcal{O}(\log t_b)$  phases and requires a communication complexity of  $\mathcal{O}(n^2\ell \log t_b)$ . However, nothing was known regarding the communication complexity of multi phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$ . The major break through in this direction was made in [77], where the authors presented the first, non trivial lower bound on the communication complexity of two phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$ . In fact, this was the first lower bound on the communication complexity which was proposed in the literature of RMT/SMT. The lower bound as given in [77] is as follows:

**Theorem 9.4 ([77])** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + 1$  wires. Then any two phase PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  must communicate  $\Omega\left(\frac{n\ell}{n-2t_b}\right)$  field elements to securely send a message containing  $\ell$  field elements.*

In another interesting work, Srinathan et al. [81] showed that the bound in Theorem 9.4 holds for any multi phase PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ . This result is stated in the following theorem:

**Theorem 9.5 ([81])** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + 1$  wires and let  $r \geq 2$ . Then any  $r$ -phase PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  must communicate  $\Omega\left(\frac{n\ell}{n-2t_b}\right)$  field elements to securely send a message containing  $\ell$  field elements.*

The significance of Theorem 9.5 is the following: it shows that *increasing the number of interactions between  $\mathbf{S}$  and  $\mathbf{R}$  does not reduce the communication complexity of PSMT protocols.*

Srinathan et al. [77] claimed that the bound in Theorem 9.4 is asymptotically tight by presenting a two phase PSMT protocol with  $n = 2t_b + 1$  wires, which securely sends  $\ell = \Theta(n)$  field elements by communicating  $\mathcal{O}(n\ell) = \mathcal{O}(n^2)$  field elements. However, in

[4], the authors showed that the two phase protocol of [77] does not provide perfect reliability. Even though the two phase protocol of [77] does not provide perfect reliability, it introduces a very novel idea, called **Matching Technique** (see Fig. 3.3), which reduces the communication complexity of reliably sending a list of conflicts. This technique was later used in several other works for designing PRMT and PSMT protocols tolerating  $\mathcal{A}_{t_b}^{static}$ .

Saurabh et al. [4] then presented a new two phase PSMT protocol with  $n = 2t_b + 1$ , tolerating  $\mathcal{A}_{t_b}^{static}$ , whose communication complexity asymptotically satisfies the bound given in Theorem 9.4. Unfortunately, in their protocol, **S** and **R** has to do exponential computation. Moreover, the protocol is communication optimal, only if the message is exponentially large. We now summarize all protocols with  $n = 2t_b + 1$  in Fig. 9.1.

Figure 9.1: Existing PSMT Protocols in Undirected Synchronous Network with  $n = 2t_b + 1$  Wires Tolerating  $\mathcal{A}_{t_b}^{static}$

Reference	Number of Phases ( $r$ )	Communication Complexity	Remark
[28]	$t_b + 1$	$\mathcal{O}(n^3\ell)$	Non optimal communication complexity
[28]	3	$\mathcal{O}(n^5\ell)$	Non optimal communication complexity
[28]	2	$\mathcal{O}(n^3\ell)$	Non optimal communication complexity <sup>a</sup>
[70]	2	$\mathcal{O}(n^3\ell)$	Non optimal communication complexity <sup>b</sup>
[85]	$\log t_b$	$\mathcal{O}(n^2\ell \log t_b)$	Non optimal communication complexity
[77]	2	$\mathcal{O}(n\ell)$	Optimal communication complexity; $\ell = \Theta(t_b)^c$
[4]	2	$\mathcal{O}(n\ell)$	Optimal communication complexity <sup>d</sup>

<sup>a</sup> In this protocol, **S** and **R** has to do exponential computation.

<sup>b</sup> By using **Matching Technique**, the communication complexity of this protocol can be made  $\mathcal{O}(n^2\ell)$ .

<sup>c</sup> In [4], the authors have shown that this protocol does not provide perfect reliability.

<sup>d</sup> In this protocol, **S** and **R** has to do exponential computation. Moreover,  $\ell$  is exponential in  $n$ .

## 9.2 A Three Phase Polynomial Time PSMT Protocol

From Fig. 9.1, we find that there does not exist any *efficient*, multi phase PSMT protocol, tolerating  $\mathcal{A}_{t_b}^{static}$  with  $n = 2t_b + 1$  wires, having optimal communication complexity. Motivated by this, we design a three phase communication optimal PSMT protocol called 3-Optimal-PSMT-Static-Byzantine with  $n = 2t_b + 1$  wires, tolerating  $\mathcal{A}_{t_b}^{static}$ . The protocol securely sends a message containing  $\ell = \Theta(n)$  field elements by communicating  $\mathcal{O}(n^2)$  field elements. Though our protocol takes one more phase than the two phase communication optimal PSMT protocol of [4], it has the following advantages: in our protocol, **S** and **R** has to do polynomial computation. Moreover, the message size is polynomial in  $n$ .

Before presenting the protocol, we present an algorithm which will be used in our protocol. The algorithm will also be used later in several other PSMT and SSMT protocols.

### 9.2.1 Extracting Randomness

Suppose  $\mathbf{S}$  and  $\mathbf{R}$  by some means agree on a sequence of  $n$  random elements  $x = (x_1, \dots, x_n) \in \mathbb{F}^n$ , such that  $\mathcal{A}_{t_b}^{static}$  knows only  $n - f$  components of  $x$ , but has no information about the other  $f$  ( $f > 0$ ) components of  $x$ . However  $\mathbf{S}$  and  $\mathbf{R}$  does not know which values are known to  $\mathcal{A}_{t_b}^{static}$ . The goal of  $\mathbf{S}$  and  $\mathbf{R}$  is to agree on a sequence of  $f$  elements  $(y_1, \dots, y_f) \in \mathbb{F}^f$ , such that  $\mathcal{A}_{t_b}^{static}$  has no information about  $(y_1, \dots, y_f)$ . This is done by executing algorithm EXTRAND, which was presented in [77]. The algorithm is presented in Fig. 9.2.

Figure 9.2: Algorithm for Extracting Randomness

**Algorithm EXTRAND** $_{n,f}(x)$ : Let  $V$  be a  $n \times f$  Vandermonde matrix with elements in  $\mathbb{F}$ . This matrix is published as a part of the algorithm specification.  $\mathbf{S}$  and  $\mathbf{R}$  both locally compute the product  $(y_1, \dots, y_f) = (x_1, \dots, x_n) \times V$ .

**Lemma 9.6** ([77]) *The  $f$  values computed in algorithm EXTRAND will be information theoretically secure.*

### 9.2.2 Protocol 3-Optimal-PSMT-Static-Byzantine

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + 1$  wires and let  $m^{\mathbf{S}}$  be the message which  $\mathbf{S}$  wants to securely send to  $\mathbf{R}$ . The message  $m^{\mathbf{S}}$  consists of  $t_b + 1 = \Theta(n)$  field elements, denoted by  $m_1^{\mathbf{S}}, \dots, m_{t_b+1}^{\mathbf{S}}$ .

The idea of protocol 3-Optimal-PSMT-Static-Byzantine is as follows:  $\mathbf{S}$  and  $\mathbf{R}$  interacts with each other to establish an information theoretic secure pad of size  $t_b + 1$ . Once this is done,  $\mathbf{S}$  can mask  $m^{\mathbf{S}}$  with the pad and reliably sends the masked message.  $\mathbf{R}$  on receiving the masked message, unmask it using the pad. In fact, this idea will be used in all the subsequent PSMT and SSMT protocols, which are going to be presented in this thesis. The only difference is how we establish the pad in the given settings.

In protocol 3-Optimal-PSMT-Static-Byzantine, the pad is established as follows: Over each wire,  $\mathbf{S}$  sends a random polynomial of degree  $t_b$  and its  $n$  values distributed over  $n$  wires.  $\mathbf{R}$  on receiving the polynomials and values perform consistency checking and constructs the conflict graph in the same way as done in Section 3.3.3.  $\mathbf{R}$  then reliably sends back the conflict graph to  $\mathbf{S}$  by using Matching Technique (see Fig. 3.3).  $\mathbf{S}$  on reliably receiving the graph identifies all wires which delivered incorrect polynomial during first phase and reliably sends their identity to  $\mathbf{R}$ . So both  $\mathbf{S}$  and  $\mathbf{R}$  will know the polynomials which are exchanged correctly. Now at least  $t_b + 1$  of these polynomials are exchanged over honest wires and  $\mathcal{A}_{t_b}^{static}$  will have no information about the constant term of these polynomials. So the constant term of these polynomials may act as the one time pad. However, neither  $\mathbf{S}$  nor  $\mathbf{R}$  knows the exact identity of the honest wires. Moreover, it may happen that some of the polynomials which are correctly exchanged are passively listened by the adversary. So  $\mathbf{S}$  and  $\mathbf{R}$  applies algorithm EXTRAND on the constant term of the polynomials which are correctly exchanged to generate a pad of size  $t_b + 1$ . The protocol is now formally presented in Fig. 9.3.

We now prove the properties of protocol 3-Optimal-PSMT-Static-Byzantine.

**Lemma 9.7 (Perfect Reliability)** *In protocol 3-Optimal-PSMT-Static-Byzantine,  $\mathbf{R}$  will be able to correctly recover  $m^{\mathbf{S}}$  at the end of Phase III.*

Figure 9.3: A Three Phase Communication Optimal PSMT Protocol Tolerating  $\mathcal{A}_{t_b}^{static}$

**Protocol 3-Optimal-PSMT-Static-Byzantine:**  $n = 2t_b + 1, m^S = (m_1^S, \dots, m_{t_b+1}^S)$

**Phase I: S to R:**

1. For  $i = 1, \dots, n$ , **S** selects a random polynomial  $p_i^S(x)$  of degree  $t_b$ .
2. **S** sends the following to **R** over wire  $w_i$ , for  $i = 1, \dots, n$ :
  - (a) Polynomial  $p_i^S(x)$ ;
  - (b) Value  $r_{ji}^S = p_j^S(i)$ , for  $j = 1, \dots, n$ .

**Phase II: R to S:**

1. Let **R** receive the following over wire  $w_i$ , for  $i = 1, \dots, n$ :
  - (a) Polynomial  $p_i^R(x)$ ;
  - (b) Value  $r_{ji}^R$ , for  $j = 1, \dots, n$ .
2. **R** constructs the conflict graph  $H$  and reliably sends it to **S** using Matching Technique <sup>a</sup>

**Phase III: S to R**

1. **S** reliably receives the conflict graph  $H$ . Thus corresponding to each edge  $(w_i, w_j)$  in  $H$ , **S** reliably receives the six-tuple  $(w_i, w_j, r_{ij}^R, p_i^R(j), r_{ji}^R, p_j^R(i))$ .
2. **S** initializes  $L_{fault}^S = L_{fault}^R$  <sup>b</sup>. Then for every received six-tuple  $(w_i, w_j, r_{ij}^R, p_i^R(j), r_{ji}^R, p_j^R(i))$ , **S** does the following computation:
  - (a) **S** checks  $r_{ij}^R \stackrel{?}{=} r_{ij}^S$  or  $p_i^R(j) \stackrel{?}{=} p_i^S(j)$ . If the first test fails then **S** concludes that  $w_j$  is corrupted and adds  $w_j$  to  $L_{fault}^S$ . On the other hand if second test fails then **S** adds  $w_i$  to  $L_{fault}^S$ .
  - (b) **S** checks  $r_{ji}^R \stackrel{?}{=} r_{ji}^S$  or  $p_j^R(i) \stackrel{?}{=} p_j^S(i)$ . If the first test fails then **S** concludes that  $w_i$  is corrupted and adds  $w_i$  to  $L_{fault}^S$ . On the other hand if the second test fails then **S** adds  $w_j$  to  $L_{fault}^S$ .
3. **S** constructs a vector  $\mathcal{P}$ , consisting of all  $p_i^S(0)$ 's, such that  $w_i \notin L_{fault}^S$ .
4. **S** computes  $Pad^S = \text{EXTRAND}_{|\mathcal{P}|, t_b+1}(\mathcal{P})$  and  $Y^S = Pad^S \oplus m^S$ .
5. **S** broadcasts  $L_{fault}^S$  and  $Y^S$  to **R** and terminates the protocol.

**Message Recovery by R**

1. **R** correctly receives  $L_{fault}^S$  and identifies all corrupted  $w_i$ 's which delivered incorrect  $p_i^S(x)$ 's during **Phase I**.
2. **R** neglects the polynomials received over the wires in  $L_{fault}^S$  during **Phase I** and computes  $Pad^S$  in the same way, as done by **S**.
3. **R** correctly receives  $Y^S$ , computes  $m^S = Y^S \oplus Pad^S$  and terminates the protocol.

<sup>a</sup> The conflict graph is constructed in the same way as done in Section 3.3.3.

<sup>b</sup> Recall that during Matching Technique, **S** receives  $L_{fault}^R$  from **R**.

PROOF: To show that **R** will be able to correctly recover  $m^S$ , it is enough to show that both **S** and **R** will agree on  $Pad^S$  at the end of **Phase III**. By the properties of Matching Technique, **S** will correctly receive the conflict graph  $H$  and will identify all corrupted  $w_i$  who has delivered incorrect  $p_i^R(x) \neq p_i^S(x)$  to **R** during **Phase I** and will include such  $w_i$ 's in  $L_{fault}^S$  (see Claim 3.12). Since **S** broadcasts  $L_{fault}^S$  during **Phase III**, **R** will also come to know the identity of such  $w_i$ 's at the end of **Phase III** and hence will neglect them. Thus at the end of **Phase III**, both **S** and **R** will agree on  $\mathcal{P}$  and hence on  $pad\ Pad^S$ .  $\square$

**Lemma 9.8 (Perfect Secrecy)** *In protocol 3-Optimal-PSMT-Static-Byzantine,  $m^{\mathbf{S}}$  will be information theoretically secure from  $\mathcal{A}_{t_b}^{\text{static}}$ , controlling at most  $t_b$  wires.*

PROOF: From the protocol description, it is easy to see that  $m^{\mathbf{S}}$  will be information theoretically secure, if the pad  $Pad^{\mathbf{S}}$  is information theoretically secure. We now show the same. First of all, notice that the set  $\mathcal{P}$  will always contain all  $p_i^{\mathbf{S}}(0)$ 's, such that  $w_i$  is an honest wire, not under the control of  $\mathcal{A}_{t_b}^{\text{static}}$ . Moreover, there will be at least  $t_b + 1$  such  $p_i^{\mathbf{S}}(0)$ 's, as there are at least  $t_b + 1$  honest wires. Furthermore,  $\mathcal{A}_{t_b}^{\text{static}}$  will have no information about the  $p_i^{\mathbf{S}}(0)$ 's delivered over the honest  $w_i$ 's. This is because  $p_i^{\mathbf{S}}(x)$  is a polynomial of degree  $t_b$  and  $\mathcal{A}_{t_b}^{\text{static}}$  gets at most  $t_b$  distinct points on  $p_i^{\mathbf{S}}(x)$  during **Phase I**. Also, if  $(w_i, w_j)$  is an edge in the conflict graph, then it implies that  $w_j$  is corrupted and hence it already knows  $p_i^{\mathbf{S}}(j)$ . This is because no two honest wire will conflict with each other. Thus throughout the protocol,  $\mathcal{A}_{t_b}^{\text{static}}$  will get only  $t_b$  distinct points on  $p_i^{\mathbf{S}}(x)$ . However, since the degree of  $p_i^{\mathbf{S}}(x)$  is  $t_b$ , the adversary will lack one point to uniquely interpolate  $p_i^{\mathbf{S}}(x)$ , implying information theoretic security for  $p_i^{\mathbf{S}}(0)$ . Thus there will be at least  $t_b + 1$   $p_i^{\mathbf{S}}(0)$ 's in  $\mathcal{P}$ , about which adversary will have no information. But  $\mathcal{P}$  may contain some  $p_i^{\mathbf{S}}(0)$ 's, such that  $w_i$  is passively controlled by the adversary. Now from the properties of algorithm EXTRAND, the pad  $Pad^{\mathbf{S}}$  will be information theoretically secure.  $\square$

**Lemma 9.9 (Communication Complexity)** *Protocol 3-Optimal-PSMT-Static-Byzantine communicates  $\mathcal{O}(n^2)$  field elements.*

PROOF: During **Phase I**,  $\mathbf{S}$  sends a polynomial of degree  $t_b$  and  $n$  values over each wire, incurring a total communication of  $\mathcal{O}(n^2)$  field elements. During **Phase II**,  $\mathbf{R}$  reliably sends conflict graph using Matching Technique. From Theorem 3.11, this requires a communication complexity of  $\mathcal{O}(n^2)$  field elements. During **Phase III**,  $\mathbf{S}$  broadcasts  $L_{\text{fault}}^{\mathbf{S}}$  and  $Y^{\mathbf{S}}$ . Since  $|L_{\text{fault}}^{\mathbf{S}}| \leq t_b$  and  $|Y^{\mathbf{S}}| = t_b + 1$ , **Phase III** will incur a communication cost of  $\mathcal{O}(n^2)$ . Thus the overall communication complexity of protocol 3-Optimal-PSMT-Static-Byzantine is  $\mathcal{O}(n^2)$ .  $\square$

**Theorem 9.10** *Protocol 3-Optimal-PSMT-Static-Byzantine is a three phase communication optimal PSMT protocol which securely sends a message containing  $\ell = \Theta(n)$  field elements by communicating  $\mathcal{O}(n^2)$  field elements.*

PROOF: The proof that 3-Optimal-PSMT-Static-Byzantine is a three phase PSMT protocol follows from Lemma 9.7 and Lemma 9.8. From Theorem 9.5, any three phase PSMT protocol over  $n = 2t_b + 1$  wires must communicate  $\Omega\left(\frac{n\ell}{n-2t_b}\right) = \Omega(n\ell)$  field elements to securely send a message containing  $\ell$  field elements against  $\mathcal{A}_{t_b}^{\text{static}}$ . Now substituting  $\ell = (t_b + 1) = \Theta(n)$ , we find that any three phase PSMT protocol over  $n = 2t_b + 1$  wires has to communicate  $\Omega(n^2)$  field elements. From Lemma 9.9, the total communication complexity of protocol 3-Optimal-PSMT-Static-Byzantine is  $\mathcal{O}(n^2)$ . Hence the protocol is communication optimal.  $\square$

### 9.3 Two Phase Communication Optimal PSMT Protocol

Though protocol 3-Optimal-PSMT-Static-Byzantine is communication optimal, it is not phase optimal. After the publication of protocol 3-Optimal-PSMT-Static-Byzantine in [58], it remained an interesting and challenging open problem to design an efficient, polynomial time, two phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{t_b}^{\text{static}}$ ,

which securely sends a message containing  $\ell$  field elements by communicating  $\mathcal{O}(n\ell)$  field elements, where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + 1$  wires. Recently, Kurosawa et al. [42] resolved this question by designing an efficient, polynomial time, two phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + 1$  wires. The protocol of Kurosawa et al. securely sends a message containing  $\ell = \Theta(n)$  field elements by communicating  $\mathcal{O}(n\ell) = \mathcal{O}(n^2)$  field elements. This completely resolved the issue of OPTIMALITY of multi phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$ .

## 9.4 Open Problem

Though the two phase PSMT protocol of [42] and our three phase PSMT protocol 3-Optimal-PSMT-Static-Byzantine are communication optimal, they are optimal only if  $\ell = \Theta(n)$ . This brings forth the following open problem:

**Open Problem 10** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + 1$  wires. Then does there exist an efficient, polynomial time multiphase PSMT protocol which securely sends a message containing  $\ell$  field elements by communicating  $\mathcal{O}(n\ell)$  field elements, tolerating  $\mathcal{A}_{t_b}^{static}$ , for any value of  $\ell$ ?*

## Chapter 10

# PSMT in Undirected Networks Against Mobile Byzantine Adversary

In this chapter, we study PSMT in *undirected synchronous network, tolerating threshold mobile Byzantine adversary*, denoted by  $\mathcal{A}_{t_b}^{mobile}$ . PSMT in undirected network, under the presence of static adversary has received quiet a bit of attention in the past [28, 70, 77, 4, 81, 30, 42]. However, as stated in Chapter 4, studying PSMT in the context of mobile adversary is well motivated. The issue of POSSIBILITY of PSMT in undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{mobile}$  is resolved in [82]. In [82], the authors have shown that connectivity requirement for PSMT in undirected synchronous network is same against  $\mathcal{A}_{t_b}^{mobile}$ , as well as  $\mathcal{A}_{t_b}^{static}$ . This shows that if the adversary does only Byzantine corruption, then mobility of the adversary does not affect the POSSIBILITY of PSMT. Since  $\mathcal{A}_{t_b}^{mobile}$  is more powerful than  $\mathcal{A}_{t_b}^{static}$ , the lower bound on the communication complexity of PSMT against  $\mathcal{A}_{t_b}^{static}$ , as given in Theorem 9.3, must hold against  $\mathcal{A}_{t_b}^{mobile}$ . This implies that if  $n = 2t_b + 1$ , then any two or more phase PSMT must communicate  $\Omega(n\ell)$  field elements to securely send a message containing  $\ell$  field elements against  $\mathcal{A}_{t_b}^{mobile}$ . Indeed, there exists a communication optimal PSMT protocol against  $\mathcal{A}_{t_b}^{mobile}$  presented in [77], which achieves the above bound and takes  $\mathcal{O}(t_b)$  phases. This shows that if the adversary does only Byzantine corruption, then mobility of the adversary does not affect the OPTIMALITY of PSMT.

In this chapter, we significantly improve the phase complexity of the communication optimal PSMT protocol of [77] tolerating  $\mathcal{A}_{t_b}^{mobile}$ . Specifically, we design a *three* phase communication optimal PSMT protocol which sends a sufficiently large message containing  $\ell$  field elements by communicating  $\mathcal{O}(n\ell)$  field elements, tolerating  $\mathcal{A}_{t_b}^{mobile}$ , where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + 1$  wires. Our three phase PSMT protocol against  $\mathcal{A}_{t_b}^{mobile}$  is different from the three phase communication optimal PSMT protocol 3-Optimal-PSMT-Static-Byzantine tolerating  $\mathcal{A}_{t_b}^{static}$ , presented in the last chapter.

### 10.1 Network Model and Adversary Settings

The network model and adversary settings used in this chapter is same as in Chapter 4. Thus, we assume that there are  $n$  bi-directional, synchronous wires  $w_1, \dots, w_n$  between  $\mathbf{S}$  and  $\mathbf{R}$ . There exists a computationally unbounded adversary  $\mathcal{A}_{t_b}^{mobile}$ , who may control *different* set of  $t_b$  wires in Byzantine fashion, during different phases of the protocol. Thus if some wire  $w_j$  is under the control of  $\mathcal{A}_{t_b}^{mobile}$  in  $i^{th}$  phase of a

protocol, then it does not imply that  $w_j$  will be corrupted in  $(i+1)^{th}$  phase also, unless  $\mathcal{A}_{t_b}^{mobile}$  controls  $w_j$  in  $(i+1)^{th}$  phase also. Moreover, by controlling  $w_j$  in  $i^{th}$  phase, the adversary  $\mathcal{A}_{t_b}^{mobile}$  will not get any information about the communication done over wire  $w_j$  in earlier phase(s), if  $\mathcal{A}_{t_b}^{mobile}$  has not controlled  $w_j$  in earlier phase(s).

## 10.2 Existing Results for PSMT Tolerating $\mathcal{A}_{t_b}^{mobile}$

Any single phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$ , will also be secure against  $\mathcal{A}_{t_b}^{mobile}$ . Thus Theorem 9.1 and Theorem 9.2 will also hold against  $\mathcal{A}_{t_b}^{mobile}$  and this completely resolves the issue of POSSIBILITY and OPTIMALITY of single phase PSMT in undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{mobile}$ .

The characterization of multi phase PSMT tolerating  $\mathcal{A}_{t_b}^{mobile}$  was given in [82], which is as follows:

**Theorem 10.1 ([82])** *Let  $r \geq 2$ . Then any  $r$ -phase PSMT protocol in undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{mobile}$  is possible iff there exists  $n \geq 2t_b + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ .*

PROOF (SKETCH): Since  $\mathcal{A}_{t_b}^{mobile}$  is more powerful than  $\mathcal{A}_{t_b}^{static}$ , the necessity follows from Theorem 9.3. To show the sufficiency, the authors in [82] showed that the two phase PSMT protocol of [70], tolerating  $\mathcal{A}_{t_b}^{static}$ , will also work against  $\mathcal{A}_{t_b}^{mobile}$ .  $\square$

Since  $\mathcal{A}_{t_b}^{mobile}$  is more powerful than  $\mathcal{A}_{t_b}^{static}$ , it implies that Theorem 9.5 must hold against  $\mathcal{A}_{t_b}^{mobile}$ . This implies that if  $n = 2t_b + 1$ , then any two or more phase PSMT must communicate  $\Omega(n\ell)$  field elements to securely send a message containing  $\ell$  field elements against  $\mathcal{A}_{t_b}^{mobile}$ . To the best of our knowledge, the only known PSMT protocol against  $\mathcal{A}_{t_b}^{mobile}$  with  $n = 2t_b + 1$  wires, achieving the above bound is due to [77]. The PSMT protocol of [77] takes  $\mathcal{O}(t_b)$  phases and securely sends a message containing  $\ell = \Theta(n)$  field element by communicating  $\mathcal{O}(n\ell) = \mathcal{O}(n^2)$  field elements.

## 10.3 A Three Phase Communication Optimal PSMT Tolerating $\mathcal{A}_{t_b}^{mobile}$

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + 1$  wires. We then present a new, communication optimal PSMT protocol called 3-Optimal-PSMT-Mobile-Byzantine, tolerating  $\mathcal{A}_{t_b}^{mobile}$ . The protocol takes three phases and securely sends a message containing  $\ell = n(t_b + 1) = \Theta(n^2)$  field elements by communicating  $\mathcal{O}(n\ell) = \mathcal{O}(n^3)$  field elements. This significantly improves the  $\mathcal{O}(t_b)$  phase communication optimal PSMT protocol of [77]. However, our protocol achieves optimality only if  $\ell = \Theta(n^2)$ , where as the PSMT protocol of [77] achieves optimality if  $\ell = \Theta(n)$ . Before proceeding further, we state the following remark.

**Remark 10.2 (3-Optimal-PSMT-Mobile-Byzantine Will Not Work Against  $\mathcal{A}_{t_b}^{mobile}$ )**  
*In the previous chapter, we have presented a three phase communication optimal PSMT protocol 3-Optimal-PSMT-Static-Byzantine, tolerating  $\mathcal{A}_{t_b}^{static}$ . However, the protocol will not work against  $\mathcal{A}_{t_b}^{mobile}$ . This is because the protocol uses Matching Technique during second phase to reliably send the conflict graph. However, as shown in Section 4.3, Matching Technique will fail to reliably send the conflict graph against  $\mathcal{A}_{t_b}^{mobile}$ .*

Protocol 3-Optimal-PSMT-Mobile-Byzantine uses the ideas presented in protocol 3-Optimal-PSMT-Static-Byzantine, along with Union Technique (see Section 4.4.1). Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + 1$  wires and let  $m^{\mathbf{S}}$  be the secret message containing  $n(t_b+1)$  field elements, denoted by  $m_{k,i}^{\mathbf{S}}$ , where  $k = 1, \dots, n$  and  $i = 1, \dots, t_b+1$ . During first phase,  $\mathbf{S}$  sends random polynomial of degree  $t_b$  over each wire and its  $n$  values across  $n$  wires, as in protocol 3-Optimal-PSMT-Static-Byzantine. However, instead of sending only one polynomial over each wire,  $\mathbf{S}$  sends  $n$  polynomials over each wire and their  $n$  values across  $n$  wires. During second phase,  $\mathbf{R}$  performs consistency checking on received values and constructs  $n$  conflict graphs and combines them into a single conflict graph using Union Technique and reliably sends the single conflict graph to  $\mathbf{S}$ . The sender  $\mathbf{S}$  then identifies all wires which have delivered at least one incorrect polynomial during first phase and reliably sends the identity of these wires to  $\mathbf{R}$ . Thus both  $\mathbf{S}$  and  $\mathbf{R}$  will now agree on the wires, which have correctly delivered all the  $n$  polynomials transmitted over them. There will be at least  $t_b + 1$  such wires.  $\mathbf{S}$  and  $\mathbf{R}$  then extracts a pad of length  $n(t_b + 1)$  and using the pad,  $\mathbf{S}$  and  $\mathbf{R}$  securely exchange the message. The protocol is formally given in Fig. 10.1.

We now prove the properties of protocol 3-Optimal-PSMT-Mobile-Byzantine.

**Lemma 10.3 (Perfect Reliability)** *In protocol 3-Optimal-PSMT-Mobile-Byzantine,  $\mathbf{R}$  will be able to correctly recover  $m^{\mathbf{S}}$  at the end of Phase III.*

PROOF: To show that  $\mathbf{R}$  will be able to correctly recover  $m^{\mathbf{S}}$ , it is enough to show that both  $\mathbf{S}$  and  $\mathbf{R}$  will agree on  $Pad^{\mathbf{S}}$  at the end of **Phase III**. By the properties of Union Technique,  $\mathbf{S}$  will correctly receive the combined conflict graph  $H^{\mathbf{R}}$  and will identify all corrupted  $w_i$  who has delivered at least one incorrect  $p_{k,i}^{\mathbf{R}}(x) \neq p_{k,i}^{\mathbf{S}}(x)$  to  $\mathbf{R}$  during **Phase I** and will include such  $w_i$ 's in  $L_{fault}$  (see Theorem 4.4). Since  $\mathbf{S}$  broadcasts  $L_{fault}$  during **Phase III**,  $\mathbf{R}$  will also come to know the identity of such  $w_i$ 's at the end of **Phase III** and hence will neglect them. Thus at the end of **Phase III**, both  $\mathbf{S}$  and  $\mathbf{R}$  will agree on  $\mathcal{P}$  and hence on pad  $Pad^{\mathbf{S}}$ .  $\square$

**Lemma 10.4 (Perfect Secrecy)** *In protocol 3-Optimal-PSMT-Mobile-Byzantine,  $m^{\mathbf{S}}$  will be information theoretically secure from  $\mathcal{A}_{t_b}^{mobile}$ , controlling at most  $t_b$  wires.*

PROOF: From the protocol description, it is easy to see that  $m^{\mathbf{S}}$  will be information theoretically secure, if the pad  $Pad^{\mathbf{S}}$  is information theoretically secure. We now show the same. First of all, notice that the set  $\mathcal{P}$  will always contain all  $p_{k,i}^{\mathbf{S}}(0)$ 's,  $k = 1, \dots, n$ , such that  $w_i$  is an honest wire during **Phase I**, not under the control of  $\mathcal{A}_{t_b}^{mobile}$ . Moreover, there will be at least  $n(t_b + 1)$  such  $p_{k,i}^{\mathbf{S}}(0)$ 's,  $k = 1, \dots, n$ , as there are at least  $t_b + 1$  honest wires  $w_i$  during **Phase I**. Furthermore,  $\mathcal{A}_{t_b}^{mobile}$  will have no information about the  $p_{k,i}^{\mathbf{S}}(0)$ 's delivered over the honest  $w_i$ 's. This is because  $p_{k,i}^{\mathbf{S}}(x)$  is a polynomial of degree  $t_b$  and  $\mathcal{A}_{t_b}^{mobile}$  gets at most  $t_b$  distinct points on  $p_{k,i}^{\mathbf{S}}(x)$  during **Phase I**. Also, if  $(w_i, w_j)$  is an edge in conflict graph  $H_k$ , then it implies that  $w_j$  is corrupted and hence it already knows  $p_{k,i}^{\mathbf{S}}(j)$ . This is because no two honest wire will conflict with each other. Thus throughout the protocol,  $\mathcal{A}_{t_b}^{mobile}$  will get only  $t_b$  distinct points on  $p_{k,i}^{\mathbf{S}}(x)$ . However, since the degree of  $p_{k,i}^{\mathbf{S}}(x)$  is  $t_b$ , the adversary will lack one point to uniquely interpolate  $p_{k,i}^{\mathbf{S}}(x)$ , implying information theoretic security for  $p_{k,i}^{\mathbf{S}}(0)$ . Thus there will be at least  $n(t_b + 1)$   $p_{k,i}^{\mathbf{S}}(0)$ 's in  $\mathcal{P}$ , about which adversary will have no information. But  $\mathcal{P}$  may contain some  $p_{k,i}^{\mathbf{S}}(0)$ 's, such that  $w_i$  is passively controlled by the adversary. Now from the properties of algorithm EXTRAND, the pad  $Pad^{\mathbf{S}}$  will be information theoretically secure.  $\square$

Figure 10.1: Three Phase Communication Optimal PSMT Tolerating  $\mathcal{A}_{t_b}^{mobile}$

<p><b>Protocol 3-Optimal-PSMT-Mobile-Byzantine:</b> <math>n = 2t_b + 1,  m^S  = n(t_b + 1)</math></p>
<p><b>Phase I: S to R:</b></p> <ol style="list-style-type: none"> <li>1. <b>S</b> selects <math>n^2</math> random polynomials of degree <math>t_b</math>, denoted by <math>p_{k,i}^S(x), k = 1, \dots, n, i = 1, \dots, n</math>.</li> <li>2. For <math>i = 1, \dots, n</math>, <b>S</b> sends the following to <b>R</b> over wire <math>w_i</math>:             <ol style="list-style-type: none"> <li>(a) Polynomials <math>p_{k,i}^S(x), k = 1, \dots, n</math>;</li> <li>(b) Values <math>r_{k,j,i}^S</math>, for <math>1 \leq k, j \leq n</math>, where <math>r_{k,j,i}^S = p_{k,j}^S(i)</math>.</li> </ol> </li> </ol>
<p><b>Phase II: R to S:</b></p> <ol style="list-style-type: none"> <li>1. Let <b>R</b> receive the following over wire <math>w_i</math>, for <math>i = 1, \dots, n</math> (see Fig. 4.1 for the pictorial representation):             <ol style="list-style-type: none"> <li>(a) Polynomials <math>p_{k,i}^R(x), k = 1, \dots, n</math>;</li> <li>(b) Values <math>r_{k,j,i}^R</math>, for <math>1 \leq k, j \leq n</math>.</li> </ol> </li> <li>2. Using the received values, <b>R</b> constructs <math>n</math> conflict graphs <math>H_1, \dots, H_n</math>. <b>R</b> then combines these graphs into a single conflict graph <math>H^R</math> using <b>Union Technique</b>, as explained in section 4.4.1.</li> <li>3. <b>R</b> constructs the list of seven tuples <math>X</math> corresponding to <math>H^R</math> (as explained in section 4.4.1) and broadcasts <math>X</math> to <b>S</b>.</li> </ol>
<p><b>Phase III: S to R:</b></p> <ol style="list-style-type: none"> <li>1. <b>S</b> reliably receives the list <math>X</math>. <b>S</b> then creates a list <math>L_{fault}</math> which is initialized to <math>\emptyset</math>.</li> <li>2. For each seven tuple <math>\{w_i, w_j, \gamma, p_{\gamma,i}^R(j), r_{\gamma,i,j}^R, p_{\gamma,j}^R(i), r_{\gamma,j,i}^R\} \in X</math>, <b>S</b> does the following:             <ol style="list-style-type: none"> <li>(a) <b>S</b> checks <math>p_{\gamma,i}^R(j) \stackrel{?}{=} p_{\gamma,i}^S(j)</math>. If not, then <b>S</b> adds <math>w_i</math> to <math>L_{fault}</math>.</li> <li>(b) <b>S</b> checks <math>r_{\gamma,i,j}^R \stackrel{?}{=} r_{\gamma,i,j}^S</math>. If not, then <b>S</b> adds <math>w_j</math> to <math>L_{fault}</math>.</li> <li>(c) <b>S</b> checks <math>p_{\gamma,j}^R(i) \stackrel{?}{=} p_{\gamma,j}^S(i)</math>. If not, then <b>S</b> adds <math>w_j</math> to <math>L_{fault}</math>.</li> <li>(d) <b>S</b> checks <math>r_{\gamma,j,i}^R \stackrel{?}{=} r_{\gamma,j,i}^S</math>. If not, then <b>S</b> adds <math>w_i</math> to <math>L_{fault}</math>.</li> </ol> </li> <li>3. <b>S</b> constructs a vector <math>\mathcal{P}</math>, consisting of all <math>p_{k,i}^S(0)</math>'s, for <math>k = 1, \dots, n</math>, such that <math>w_i \notin L_{fault}</math>.</li> <li>4. <b>S</b> computes <math>Pad^S = \text{EXTRAND}_{ \mathcal{P} , n(t_b+1)}(\mathcal{P})</math> and <math>Y^S = Pad^S \oplus m^S</math>.</li> <li>5. <b>S</b> broadcasts <math>L_{fault}</math> and <math>Y^S</math> to <b>R</b> and terminates 3-Optimal-PSMT-Mobile-Byzantine.</li> </ol>
<p><b>Message Recovery by R:</b></p> <ol style="list-style-type: none"> <li>1. <b>R</b> reliably receives <math>L_{fault}</math> and identifies all <math>w_i</math> over which it had received at least one incorrect polynomial <math>p_{k,i}^R(x) \neq p_{k,i}^S(x)</math> during <b>Phase I</b>.</li> <li>2. <b>R</b> neglects all the polynomials <math>p_{k,i}^R(x), k = 1, \dots, n</math>, received over each <math>w_i \in L_{fault}</math> during <b>Phase I</b>.</li> <li>3. <b>R</b> computes <math>Pad^S</math> in the same way, as done by <b>S</b>.</li> <li>4. <b>R</b> correctly receives <math>Y^S</math>, computes <math>m^S = Y^S \oplus Pad^S</math> and terminates the protocol.</li> </ol>

**Lemma 10.5 (Communication Complexity) Protocol**

*3-Optimal-PSMT-Mobile-Byzantine communicates  $\mathcal{O}(n^3)$  field elements.*

PROOF: During **Phase I**, **S** sends  $n$  polynomials of degree  $t_b$  and  $n^2$  values over each wire, incurring a total communication of  $\mathcal{O}(n^3)$  field elements. During **Phase II**, **R** broadcasts the list  $X$ . As explained in Section 4.4.1,  $X$  contains  $\mathcal{O}(n^2)$  tuples. Hence broadcasting  $X$  requires  $\mathcal{O}(n^3)$  communication complexity. During **Phase III**, **S** broadcasts  $L_{fault}$  and  $Y^S$ . Since  $|L_{fault}| \leq t_b$  and  $|Y^S| = n(t_b + 1)$ , **Phase III** will

incur a communication cost of  $\mathcal{O}(n^3)$ . Thus the overall communication complexity of protocol 3-Optimal-PSMT-Mobile-Byzantine is  $\mathcal{O}(n^3)$ .  $\square$

**Theorem 10.6** *Protocol 3-Optimal-PSMT-Mobile-Byzantine is a three phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{t_b}^{mobile}$ , which securely sends a message containing  $\ell = \Theta(n^2)$  field elements by communicating  $\mathcal{O}(n^3)$  field elements.*

PROOF: The proof that 3-Optimal-PSMT-Mobile-Byzantine is a three phase PSMT protocol tolerating  $\mathcal{A}_{t_b}^{mobile}$  follows from Lemma 10.3 and Lemma 10.4. As mentioned earlier, Theorem 9.5 will also hold against  $\mathcal{A}_{t_b}^{mobile}$  and hence any three phase PSMT protocol over  $n = 2t_b + 1$  wires must communicate  $\Omega\left(\frac{n\ell}{n-2t_b}\right) = \Omega(n\ell)$  field elements to securely send a message containing  $\ell$  field elements against  $\mathcal{A}_{t_b}^{mobile}$ . Now substituting  $\ell = n(t_b + 1) = \Theta(n^2)$ , we find that any three phase PSMT protocol over  $n = 2t_b + 1$  wires has to communicate  $\Omega(n^3)$  field elements. From Lemma 10.5, the total communication complexity of protocol 3-Optimal-PSMT-Mobile-Byzantine is  $\mathcal{O}(n^3)$ . Hence the protocol is communication optimal.  $\square$

## 10.4 Concluding Remarks and Open Problems

In this chapter, we presented a three phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{t_b}^{mobile}$ , thus significantly improving the  $\mathcal{O}(t_b)$  phase communication optimal PSMT protocol of [77]. As mentioned in the last chapter, Kurosawa et al. presented a two phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , which securely sends a message containing  $\ell = \Theta(n)$  field elements by communicating  $\mathcal{O}(n\ell) = \mathcal{O}(n^2)$  field elements, where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + 1$  wires. However, the protocol will not work against  $\mathcal{A}_{t_b}^{mobile}$  because the protocol uses **Matching Technique** and as explained in Section 4.3, the **Matching Technique** will not work against  $\mathcal{A}_{t_b}^{mobile}$ . This brings forth a very interesting and challenging open problem, which is as follows:

**Open Problem 11** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + 1$  wires. Then does there exist a two phase PSMT protocol which securely sends a message containing  $\ell$  field elements, by communicating  $\mathcal{O}(n\ell)$  field elements, tolerating  $\mathcal{A}_{t_b}^{mobile}$ , for any value of  $\ell$ ?*

## Chapter 11

# PSMT in Undirected Networks Tolerating Static Mixed Adversary

In this chapter, we study PSMT in *undirected synchronous network, tolerating threshold static mixed adversary*. The static mixed adversary, denoted by  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , corrupts disjoint set of  $t_b, t_f$  and  $t_p$  nodes in Byzantine, failstop and passive fashion respectively. Studying PSMT in the context of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is well motivated. In a real life network, the adversary may do different type of corruptions. For example, a weakly protected node may be Byzantine corrupted, while a strongly protected node may be passively or failstop corrupted.

The characterization for single phase and multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is given in [75]. In [75], the author has also derived the lower bound on the communication complexity of single and multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . Moreover, the author also presented a single phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , thus completely resolving the issue of POSSIBILITY and OPTIMALITY of single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . However, no multi phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  was presented. This left the problem of designing multi phase communication optimal PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  as open. In this chapter, we settle this problem by designing a four phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . Interestingly, we find that the techniques used to design our three phase communication optimal PSMT protocol against  $\mathcal{A}_{t_b}^{static}$  in Chapter 9 cannot be extended in a straight forward manner to design communication optimal PSMT protocol against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . Hence to design our PSMT protocol against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , we use some different techniques. We now describe the network model and adversary settings used in this chapter.

### 11.1 Network Model and Adversary Settings

The underlying network model is similar as in Chapter 5. However, instead of  $\mathcal{A}_{(t_b, t_f)}^{static}$ , we assume the presence of mixed adversary  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . Thus, we assume that  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n$  bidirectional, synchronous wires  $w_1, \dots, w_n$ , of which at most  $t_b, t_f$  and  $t_p$  wires can be under the control of a computationally unbounded  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  in Byzantine, failstop and passive fashion respectively. Moreover, once a wire is under the

control of the adversary is some fashion, it will remain so for the rest of the protocol. Furthermore, as a worst case assumption, we assume that the wires that are under the control of the adversary in Byzantine, failstop and passive fashion are mutually disjoint.

The characteristic of Byzantine corrupted and failstop corrupted wire is already specified in Chapter 5. A wire which is controlled in passive fashion always deliver correct information. However, the adversary  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  will also know the complete information which is delivered through a passively controlled wire. Since a Byzantine corrupted wire is also a passively corrupted wire, it implies that  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  can eavesdrop at most  $t_b + t_p$  wires throughout the protocol.

We now recall the existing results for PSMT in the presence of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  from [75].

## 11.2 Existing Results for PSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

We first recall the existing results for single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .

### 11.2.1 Characterization and Lower Bound for Single Phase PSMT

The PSMT problem in the presence of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  was first studied in [75]. In [75], Srinathan gave the following necessary and sufficient condition for the existence of single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .

**Theorem 11.1 ([75])** *Any single phase PSMT in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is possible iff there exists  $n \geq 3t_b + t_f + t_p + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ .*

PROOF (SKETCH): The necessity is proved by showing that in any single phase PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , the information sent by  $\mathbf{S}$  over any  $n - (2t_b + t_f)$  wires should have full information about the message. Since in any single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  the adversary can passively listen at most  $t_b + t_p$  wires, the above condition implies that in any single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ ,  $n - (2t_b + t_f) > (t_b + t_p)$  should hold, otherwise adversary will also know the secret message. This further implies that  $n > (3t_b + t_f + t_p)$  should hold.

The sufficiency is shown as follows: suppose there exists  $n = 3t_b + t_f + t_p + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ . To securely send a message  $m^{\mathbf{S}}$ , sender  $\mathbf{S}$  selects a random polynomial of degree  $t_b + t_p$ , whose constant term is  $m^{\mathbf{S}}$  and computes an RS codeword of length  $n$  from this polynomial.  $\mathbf{S}$  then sends one component of the codeword to  $\mathbf{R}$  over each wire.  $\mathbf{R}$  will receive at least  $3t_b + t_p + 1$  points on a polynomial of degree  $t_b + t_p$  and at most  $t_b$  points could be corrupted. However, by applying RS-DEC,  $\mathbf{R}$  will be able to correct these errors and recover the original polynomial and hence the message  $m^{\mathbf{S}}$ . The secrecy of  $m^{\mathbf{S}}$  follows from the fact that  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  will know only  $t_b + t_p$  points on the polynomial, whose degree is  $t_b + t_p$ . Thus adversary will lack by one point to uniquely interpolate the polynomial, implying information theoretic security for  $m^{\mathbf{S}}$ .  $\square$

**Comparison 11.2 (Theorem 9.1 and Theorem 11.1)** *The significance of Theorem 11.1 over Theorem 9.1 is established by the following two facts:*

1. *Theorem 11.1 generalizes Theorem 9.1 as we get the later by substituting  $t_f = t_p = 0$  in the former.*

2. Theorem 11.1 shows the availability of more fault tolerance in comparison to Theorem 9.1. For a clean interpretation of this statement, consider a network with five wires between **S** and **R**. From Theorem 9.1, the network can tolerate only one Byzantine corruption. However, from Theorem 11.1, the network can tolerate one Byzantine corruption, along with one additional fault, which can be either passive or fail-stop type. This example clearly justifies the need to study PSMT in the context of mixed adversary. Had we treated the passive or fail-stop corruption as Byzantine corruption, we would require seven wires between **S** and **R** (from Theorem 9.1), which is much more than what is actually required.

In [75], the author gave the following lower bound on the communication complexity of single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ :

**Theorem 11.3 ([75])** *Let **S** and **R** be connected by  $n \geq 3t_b + t_f + t_p + 1$  wires. Then any single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  must communicate  $\Omega\left(\frac{n\ell}{n-(3t_b+t_f+t_p)}\right)$  field elements to securely send a message containing  $\ell$  field elements. Moreover, this bound is tight.*

PROOF (SKETCH): The lower bound is derived by first showing that the communication complexity of any single phase PSMT to securely send a message  $m^{\mathbf{S}}$  tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is not less than the share complexity (sum of the length of the shares) of a secret sharing scheme, which generates  $n$  shares for  $m^{\mathbf{S}}$ , such that any set of  $n - (2t_b + t_f)$  shares has full information about the message, while any set of  $t_b + t_p$  shares has no information about the message. Then it is shown that the share complexity (sum of the length of the shares) of any secret sharing scheme, which generates  $n$  shares for a message  $m^{\mathbf{S}}$  containing  $\ell$  field elements, such that any set of  $n - (2t_b + t_f)$  shares has full information about the message, while any set of  $t_b + t_p$  shares has no information about the message is  $\Omega\left(\frac{n\ell}{n-(3t_b+t_f+t_p)}\right)$ . This is done by using similar arguments as given in Lemma 6.11.

The tightness of the bound follows from the protocol which is sketched in Theorem 11.1, which securely sends a message containing  $\ell = 1$  field element by communicating  $\mathcal{O}(n)$  field elements, where **S** and **R** are connected by  $n = 3t_b + t_f + t_p + 1$  wires.  $\square$

In the next section, we recall the existing results for multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .

### 11.2.2 Characterization and Lower Bound for Multi Phase PSMT

In [75], Srinathan gave the following characterization for multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .

**Theorem 11.4 ([75])** *Let  $r \geq 2$ . Then any  $r$ -phase PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is possible iff there exists  $n \geq 2t_b + t_f + t_p + 1$  wires between **S** and **R**.*

PROOF (SKETCH): The necessity is proved by showing that if there exists an  $r$ -phase PSMT in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  with  $n = 2t_b + t_f + t_p$  wires between **S** and **R**, where  $r \geq 2$ , then there exists an  $r$ -phase PSMT in undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{static}$  with  $n = 2t_b$  wires between **S** and **R**, which from Theorem 9.3 is not possible.

To show the sufficiency, the author in [75] presented a two phase PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , with  $n = 2t_b + t_f + t_p + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ . Specifically, the author showed how to extend the two phase PSMT protocol of [70] with  $n = 2t_b + 1$  wires tolerating  $\mathcal{A}_{t_b}^{static}$ , to tolerate  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  over  $n = 2t_b + t_f + t_p + 1$  wires.  $\square$

In [75], the author gave the following lower bound on the communication complexity of multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .

**Theorem 11.5 ([75])** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + t_f + t_p + 1$  wires and let  $r \geq 2$ . Then any  $r$ -phase PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  must communicate  $\Omega\left(\frac{n\ell}{n-(2t_b+t_f+t_p)}\right)$  field elements to securely send a message containing  $\ell$  field elements.*

To the best of our knowledge, no communication optimal multi phase PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is known, which satisfies the bound given in Theorem 11.5. We make inroads towards this by presenting a four phase communication optimal PSMT protocol called 4-Optimal-PSMT-Static-Mixed, tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + t_f + t_p + 1$  wires. Before presenting the protocol, we show why the three phase communication optimal PSMT protocol 3-Optimal-PSMT-Static-Byzantine tolerating  $\mathcal{A}_{t_b}^{static}$ , cannot be extended in a straight forward manner to tolerate  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .

### 11.3 Limitations of Protocol 3-Optimal-PSMT-Static-Byzantine Against $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

In Fig. 9.3, we presented a three phase communication optimal PSMT protocol called 3-Optimal-PSMT-Static-Byzantine, tolerating  $\mathcal{A}_{t_b}^{static}$ , where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + 1$  wires. The protocol securely sends a message containing  $\ell = (t_b + 1) = \Theta(n)$  field elements by communicating  $\mathcal{O}(n\ell) = \mathcal{O}(n^2)$  field elements. Recall that the underlying idea of protocol 3-Optimal-PSMT-Static-Byzantine is as follows:  $\mathbf{S}$  sends one random  $t_b$  degree polynomial over each of the  $n$  wires and their  $n$  values distributed over  $n$  wires. After a sequence of interaction between  $\mathbf{S}$  and  $\mathbf{R}$  according to the protocol, the constant coefficients of the  $t_b + 1$  polynomials which are not under the control of the adversary, are established as an *information theoretic secure* "one time pad" between  $\mathbf{S}$  and  $\mathbf{R}$ . Moreover the communication complexity of the interaction is  $\mathcal{O}(n^2)$ . Now using this one time pad,  $\mathbf{S}$  securely sends  $t_b + 1 = \Theta(n)$  field elements to  $\mathbf{R}$  by communicating  $\mathcal{O}(n^2)$  field elements.

For tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ ,  $\mathbf{S}$  and  $\mathbf{R}$  must be connected by at least  $n = 2t_b + t_f + t_p + 1$  wires. Now if we use the same technique of sending polynomials as well as their values (as used in protocol 3-Optimal-PSMT-Static-Byzantine), then  $\mathbf{S}$  and  $\mathbf{R}$  ends up in establishing a secure "one time pad" of length  $t_b + 1$  after communicating  $\mathcal{O}(n^2)$  field elements. The reason is that  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  can failstop  $t_f$  wires and passively listen the polynomials over  $(t_b + t_p)$  wires. Therefore only  $n - t_f - t_b - t_p = t_b + 1$  polynomials will be unknown to the adversary. Since  $n = 2t_b + t_f + t_p + 1$ ,  $t_b$  may not be  $\Theta(n)$  and can even be a constant. Thus the resulting PSMT protocol may send a message of very small size with very high communication complexity of  $\mathcal{O}(n^2)$ , which will not be a communication optimal PSMT protocol against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .

Hence the techniques of protocol 3-Optimal-PSMT-Static-Byzantine for designing communication optimal PSMT protocol against  $\mathcal{A}_{t_b}^{static}$  cannot be extended in a straight-forward manner for designing communication optimal PSMT protocol against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .

To achieve optimality,  $\mathbf{S}$  should be able to securely establish a one time pad of size  $t_b + t_f + t_p$ , instead of  $t_b + 1$  by communicating  $\mathcal{O}(n^2)$  field elements. Thus we require new techniques for designing communication optimal PSMT protocol against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , which we explore in the sequel.

## 11.4 A Four Phase Communication Optimal PSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + t_f + t_p + 1$  wires  $w_i, 1 \leq i \leq n$ . We design a four phase communication optimal PSMT protocol called 4-Optimal-PSMT-Static-Mixed, which securely sends a message  $m^{\mathbf{S}}$  containing  $n$  field elements by communicating  $\mathcal{O}(n^2)$  field elements, tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . We first design few sub-protocols and finally combine them to obtain 4-Optimal-PSMT-Static-Mixed.

### 11.4.1 A Conditional Single Phase Protocol to Establish a One Time Pad

Suppose  $\mathbf{A}$  and  $\mathbf{B}$  are connected by  $n = 2t_b + t_f + t_p + 1$  wires that are under the influence of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . Here  $\mathbf{A}$  can be  $\mathbf{S}$  ( $\mathbf{R}$ ) and  $\mathbf{B}$  can be  $\mathbf{R}$  ( $\mathbf{S}$ ). Also assume that  $\mathbf{A}$  in advance knows the identity of at least  $\frac{t_b}{2}$  wires which are under the control of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  in Byzantine fashion. Under this assumption, we design a single phase protocol called Pad-Establishment-Static, which securely establishes a random one time pad of length  $n$  between  $\mathbf{A}$  and  $\mathbf{B}$ , which is information theoretically secure from  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . The protocol is given in Fig. 11.1.

We now prove the properties of protocol Pad-Establishment-Static.

**Lemma 11.6 (Correctness)** *Suppose  $\mathbf{A}$  in advance knows the identity of at least  $\frac{t_b}{2}$  wires which are under the control of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  in Byzantine fashion. Then protocol Pad-Establishment-Static correctly establishes the  $n$  tuple  $q = (q_1^{\mathbf{A}}(0), \dots, q_n^{\mathbf{A}}(0))$  between  $\mathbf{A}$  and  $\mathbf{B}$  in a single phase, tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .*

PROOF: From  $L_{fault}$ ,  $\mathbf{B}$  identifies  $|L_{fault}| \geq \frac{t_b}{2}$  Byzantine corrupted wires and neglects them. Among the remaining wires, at most  $t_f$  may fail to deliver any information due to fail-stop corruption. So in the worst case,  $N' = n - |L_{fault}| - t_f$ . In the protocol, each  $Q_j^{\mathbf{A}}$  is RS encoded using a polynomial of degree  $t_b - |L_{fault}| + t_p$ . Since  $\mathbf{B}$  has neglected information corresponding to  $|L_{fault}|$  Byzantine corrupted wires, the received vector  $Q_j^{\mathbf{B}}$  will differ from the original RS codeword  $Q_j^{\mathbf{A}}$  in at most  $t_b - |L_{fault}|$  locations. Substituting  $N' = n - |L_{fault}| - t_f, k = t_b - |L_{fault}| + t_p + 1, c = t_b - |L_{fault}|$  and  $d = 0$  in the inequality of Theorem 2.19, we find that RS-DEC( $N', Q_j^{\mathbf{B}}, t_b - |L_{fault}|, 0, t_b - |L_{fault}| + t_p + 1$ ) will be able to correct all the  $t_b - |L_{fault}| \leq \frac{t_b}{2}$  Byzantine errors in  $Q_j^{\mathbf{B}}$ . Thus  $\mathbf{B}$  can recover the original polynomial  $q_j^{\mathbf{A}}(x)$  and hence  $q_j^{\mathbf{A}}(0)$ .  $\square$

**Lemma 11.7 (Security)** *In protocol Pad-Establishment-Static,  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  will get no information about the pad  $q = (q_1^{\mathbf{A}}(0), \dots, q_n^{\mathbf{A}}(0))$ .*

PROOF: In the protocol, the adversary gets at most  $t_b - |L_{fault}| + t_p$  distinct points on each polynomial  $q_j^{\mathbf{A}}(x)$  of degree  $t_b - |L_{fault}| + t_p$ . This implies information theoretic security of each  $q_j^{\mathbf{A}}(0)$ .  $\square$

**Lemma 11.8 (Communication Complexity)** *Protocol Pad-Establishment-Static communicates  $\mathcal{O}(n^2)$  field elements.*

Figure 11.1: Single Phase Protocol to Establish a One Time Pad of length  $n = 2t_b + t_f + t_p + 1$

**Protocol Pad-Establishment-Static:**  $n = 2t_b + t_f + t_p + 1$

**Assumption:** **A** knows the identity of at least  $\frac{t_b}{2}$  wires, which are Byzantine corrupted.

**Computation by A :**

1. **A** saves the identity of the wires which are known to be Byzantine corrupted, in a list  $L_{fault}$ . According to the assumption,  $\frac{t_b}{2} \leq |L_{fault}| \leq t_b$ .
2. **A** selects  $n$  random polynomials  $q_j^A(x), 1 \leq j \leq n$ , over  $\mathbb{F}$ , each of degree  $t_b - |L_{fault}| + t_p$ .
3. For  $j = 1, \dots, n$ , using  $q_j^A(x)$ , **A** computes an RS codeword  $Q_j^A = (q_{j1}^A, \dots, q_{jn}^A)$  of length  $n$ , where  $q_{ji}^A = q_j^A(i)$ , for  $i = 1, \dots, n$ .

**Communication by A :**

1. For  $i = 1, \dots, n$ , if  $w_i \notin L_{fault}$ , then **A** sends to **B** the  $i^{th}$  component of the  $n$  codewords, namely  $q_{i1}^A, \dots, q_{in}^A$  over wire  $w_i$ .
2. **A** broadcasts  $L_{fault}$  to **B**.

**Computation by B :**

1. **B** correctly receives  $L_{fault}$  and neglects any information received over  $w_i \in L_{fault}$ .
2. Among the wires in  $\{w_1, \dots, w_n\} - L_{fault}$ , let  $w_{i_1}, \dots, w_{i_{N'}}$  be the wires that delivered information to **B**. Note that  $n - t_f - |L_{fault}| \leq N' \leq n - |L_{fault}|$ . Moreover,  $N'$  is at least  $n - t_f - |L_{fault}|$ , as among the  $n - |L_{fault}|$  wires, at most  $t_f$  wires may fail to deliver any information due to fail-stop corruption.
3. For  $j = 1, \dots, n$ , let **B** receive  $q_{ji_1}^B, \dots, q_{ji_{N'}}^B$  over wires  $w_{i_1}, \dots, w_{i_{N'}}$ , respectively. Let  $Q_j^B = (q_{ji_1}^B, \dots, q_{ji_{N'}}^B)$ .
4. For  $j = 1, \dots, n$ , **B** recovers  $q_j^A(x)$  (and hence  $q_j^A(0)$ ) by executing RS-DEC( $N', Q_j^B, t_b - |L_{fault}|, 0, t_b - |L_{fault}| + t_p + 1$ ).

The  $n$  tuple  $q = (q_1^A(0), \dots, q_n^A(0))$  is established correctly and securely between **A** and **B**.

PROOF: For each  $q_j^A(x), 1 \leq j \leq n$ , **A** sends  $n - |L_{fault}| = \mathcal{O}(n)$  values which incurs a total communication complexity of  $\mathcal{O}(n^2)$ . Also communication complexity of broadcasting  $L_{fault}$  is  $\mathcal{O}(n^2)$ .  $\square$

### 11.4.2 A Three Phase Protocol to Identify at least $\frac{t_b}{2}$ Byzantine Corrupted Wires

As before, let **A** and **B** are connected by  $n = 2t_b + t_f + t_p + 1$  wires. We now design a three phase protocol called **Error-Identification-Static** that has the following properties:

1. If at most  $\frac{t_b}{2}$  wires get Byzantine corrupted during first phase, then **A** securely establishes a one time pad of length  $n$  with **B** at the end of second phase.
2. If more than  $\frac{t_b}{2}$  wires get Byzantine corrupted during first phase, then the pad will not be established. However, either **A** comes to know the identity of at least  $\frac{t_b}{2}$  Byzantine corrupted wires at the end of second phase or **B** comes to know the identity of at least  $\frac{t_b}{2}$  Byzantine corrupted wires at the end of third phase, depending upon the adversary behavior.

Thus the protocol creates a win-win situation against the adversary as follows: if the adversary does at most  $\frac{t_b}{2}$  Byzantine faults in the first phase, then an information theoretic secure one time pad is established between **A** and **B**. Otherwise, either **A** or **B** will come to know the identity of more than  $\frac{t_b}{2}$  Byzantine corrupted wires.

Informally, the protocol works as follows: **A** selects  $n$  polynomials each of degree  $t_b + t_p$  and sends an RS codeword of length  $n$  for each of these polynomials to **B**. **B** applies RS-DEC to the received vectors, assuming the number of errors in the vectors to be at most  $\frac{t_b}{2}$  and tries to recover the  $n$  polynomials. If corresponding to some vector, **B** is unable to recover anything, then **B** concludes that more than  $\frac{t_b}{2}$  errors occurred in that vector. In this case, **B** sends back this vector to **A**, who after comparing it with its corresponding original codeword, finds the identity of at least  $\frac{t_b}{2} + 1$  corrupted wires. Otherwise, **B** recovers  $n$  polynomials of degree  $t_b + t_p$ , but is unable to decide about their correctness. In this case, **B** broadcasts the  $n$  error lists to **A**, that are output by applying RS-DEC to the received vectors. **A** then verifies whether all the error lists are “good” or not. If yes, then **A** concludes that **B** has correctly recovered all the  $n$  polynomials. Otherwise, **A** identifies at least one polynomial that is *not* recovered correctly by **B** because of more than  $\frac{t_b}{2}$  faults during **Phase I**. **A** then broadcasts to **B**, the original codeword of that polynomial, generated by him during **Phase I**. This broadcast enables **B** to identify more than  $\frac{t_b}{2}$  faults after local verification at the end of **Phase III**. The protocol is now formally presented in Fig. 11.2.

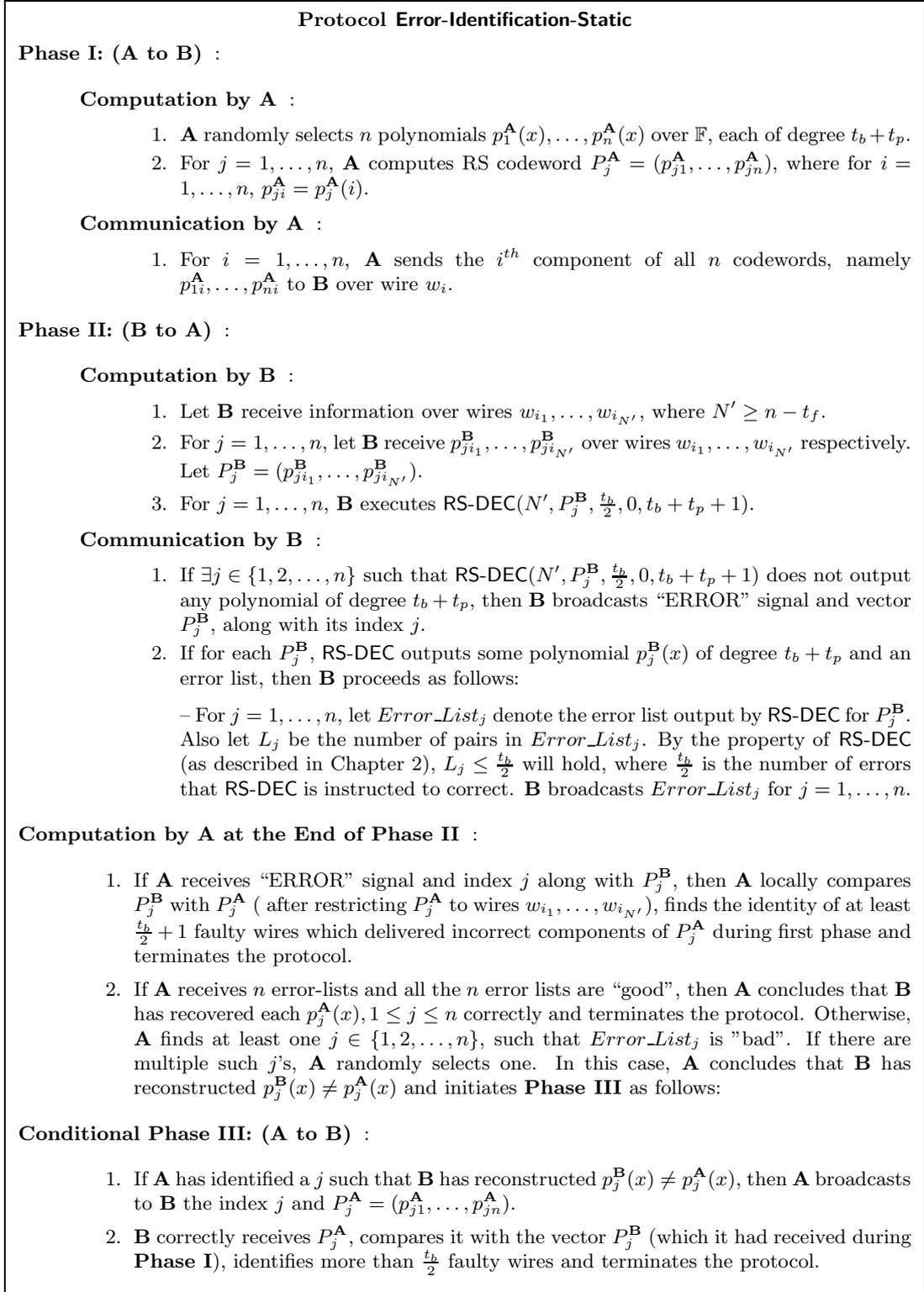
We now proceed to formally prove the properties of protocol **Error-Identification-Static**.

**Lemma 11.9** *In protocol Error-Identification-Static:*

1. If at most  $\frac{t_b}{2}$  Byzantine errors occur during **Phase I**, then an information theoretically secure pad  $p = (p_1^A(0), \dots, p_n^A(0))$  of length  $n$  is established between **A** and **B** at the end of **Phase II**.
2. If more than  $\frac{t_b}{2}$  Byzantine errors occur during **Phase I**, then either **A** or **B** comes to know the identity of more than  $\frac{t_b}{2}$  corrupted wires at the end of **Phase II** or **Phase III** respectively.

*Proof.* We prove the theorem for the worst case where during **Phase I**,  $t_f$  wires failed to deliver any information to **B**. Thus **B** receives information over  $N' = n - t_f = 2t_b + t_p + 1$  wires during first phase. For  $j = 1, \dots, n$ , each of the received vector  $P_j^B$  will contain  $N' = 2t_b + t_p + 1$  values, out of which at most  $t_b$  could be corrupted. Also, each  $P_j^B$  corresponds to the RS codeword  $P_j^A$  that is RS encoded using polynomial  $p_j^A(x)$  of

Figure 11.2: A Three Phase Protocol to Identify More than  $\frac{t_b}{2}$  Byzantine Faults



degree  $t_b + t_p$ . During **Phase II**, **B** tries to correct at most  $c = \frac{t_b}{2}$  and detect additional  $d = 0$  errors in each  $P_j^{\mathbf{B}}$  by applying RS-DEC. By substituting  $N' = 2t_b + t_p + 1$ ,  $c = \frac{t_b}{2}$ ,  $d = 0$  and  $k = t_b + t_p + 1$  in the inequality of Theorem 2.19, we find that RS-DEC( $N', P_j^{\mathbf{B}}, \frac{t_b}{2}, 0, t_b + t_p + 1$ ) will be able to correct at most  $\frac{t_b}{2}$  errors and detect *no* additional errors in  $P_j^{\mathbf{B}}$ . Moreover, **B** has no information about the exact number of Byzantine errors that occurred during **Phase I** (except that it is at most  $t_b$ ). Now there are following two cases:

1. **Case I: At most  $\frac{t_b}{2}$  Byzantine errors occurred during Phase I:** In this case at most  $\frac{t_b}{2}$  values in each  $P_j^{\mathbf{B}}$  could be corrupted. Hence for each  $P_j^{\mathbf{B}}$ , RS-DEC will output  $p_j^{\mathbf{B}}(x) = p^{\mathbf{A}}(x)$  and a "good" error list after successfully correcting all the errors in  $P_j^{\mathbf{B}}$ . However, **B** will not know whether the recovered polynomials are correct or not. This is because the number of actual errors that can happen (i.e.  $t'_b$ ) can be more than the error correction capability (i.e.  $c$ ) of RS-DEC. Moreover, as RS-DEC has no capability of detecting additional errors (as  $d = 0$  here), **B** is not sure whether  $p_j^{\mathbf{B}}(x)$  is the original polynomial used for encoding  $P_j^{\mathbf{B}}$  and the error list is "good". The situation here is similar to the one that arises in Example 2.16. Hence to know the status of the recovered polynomials, **B** broadcasts each error list to **A** who will correctly receive them <sup>1</sup>. When **A** gets the error lists from **B** and finds them to be "good", he concludes that **B** has recovered each  $p_j^{\mathbf{A}}(x)$  correctly. Hence the vector  $p = (p_1^{\mathbf{A}}(0), \dots, p_n^{\mathbf{A}}(0))$  is established correctly between **A** and **B**.

The security of  $p$  follows from the fact that during **Phase I**, the adversary gets at most  $t_b + t_p$  points (by passively listening over  $t_b + t_p$  wires) on each  $p_j^{\mathbf{A}}(x)$ , which is of degree  $t_b + t_p$ . Thus each  $p_j^{\mathbf{A}}(0)$  is information theoretically secure. Also notice that each of the  $n$  error lists are "good", thus they leak no extra information about  $p_j^{\mathbf{A}}(x)$ 's to  $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{static}}$ . This is because in this case, the values in each error lists are indeed corrupted, which are already known to the adversary and hence add no extra information to the knowledge of adversary.

2. **Case II: More than  $\frac{t_b}{2}$  Byzantine errors occurred during Phase I:** Without loss of generality, let  $p_j^{\mathbf{A}}(x)$  be one of the polynomials, corresponding to which at least  $\frac{t_b}{2} + 1$  values have been corrupted by adversary during **Phase I**. Thus  $j^{\text{th}}$  received vector  $P_j^{\mathbf{B}}$  will have more than  $\frac{t_b}{2}$  corrupted values. So **B** will fail to correctly reconstruct  $p_j^{\mathbf{A}}(x)$  by executing RS-DEC( $N', P_j^{\mathbf{B}}, \frac{t_b}{2}, 0, t_b + t_p + 1$ ). Now there are two possible cases:

- (a) Suppose the values in  $P_j^{\mathbf{B}}$  are corrupted in such a way that RS-DEC, when applied to  $P_j^{\mathbf{B}}$ , fails to output any  $t_b + t_p$  degree polynomial. In this case, **B** knows that more than  $\frac{t_b}{2}$  values in  $P_j^{\mathbf{B}}$  are corrupted. However, he will not know the exact identity of the corrupted wires, who delivered those corrupted values. In order to facilitate **A** to find the identity of those corrupted wires, **B** broadcasts  $P_j^{\mathbf{B}}$  to **A**, along with "ERROR" signal and index  $j$ . Once **A** correctly receives these values and performs local comparison of  $P_j^{\mathbf{B}}$  with  $P_j^{\mathbf{A}}$ , **A** will know the identity of all the corrupted wires (at least  $\frac{t_b}{2} + 1$ ) who delivered incorrect components of  $P_j^{\mathbf{A}}$  to **B** during first phase.

---

<sup>1</sup>Any information broadcasted over  $n = 2t_b + t_f + t_p + 1$  will be received correctly at the receiving end by taking majority, irrespective of adversary behavior.

- (b) Suppose the values in  $P_j^{\mathbf{B}}$  are corrupted in such a way that RS-DEC, when applied to  $P_j^{\mathbf{B}}$ , outputs a  $t_b+t_p$  degree polynomial  $p_j^{\mathbf{B}}(x)$ , along with  $Error\_List_j$ . In this case  $p_j^{\mathbf{B}}(x) \neq p_j^{\mathbf{A}}(x)$ . Moreover,  $Error\_List_j$  is "bad" and contains a correct component of  $P_j^{\mathbf{A}}$  in it (pointed as a corrupted value)<sup>2</sup>. The reason is that there are  $t_b+t_p+1$  wires, which will deliver correct points on  $p_j^{\mathbf{A}}(x)$ . Among them at most  $t_b+t_p$  could lie on  $p_j^{\mathbf{B}}(x)$  as well, as  $p_j^{\mathbf{A}}(x)$  and  $p_j^{\mathbf{B}}(x)$  (both of degree  $t_b+t_p$ ) may have at most  $t_b+t_p$  common points lying on them. But the remaining one honest value lies on *only*  $p_j^{\mathbf{A}}(x)$  and not lies on  $p_j^{\mathbf{B}}(x)$ . Hence that honest value will be considered as an error location and will be included in  $Error\_List_j$ .

However as described in **Case I**, **B** will not know whether the recovered polynomial is correct or not. Hence, **B** broadcasts  $Error\_List_j$  to **A**. Once **A** correctly receives  $Error\_List_j$  and performs local verification, **A** will find  $Error\_List_j$  to be "bad" and will conclude that **B** has recovered incorrect  $p_j^{\mathbf{A}}(x)$  because of more than  $\frac{t_b}{2}$  incorrect values in  $P_j^{\mathbf{B}}$ . But **A** will not know the identity of these corrupted wires. To facilitate **B** to find out the identity of corrupted wires, **A** will execute third phase, where he will broadcast  $P_j^{\mathbf{A}}$  to **B**. After receiving  $P_j^{\mathbf{A}}$  correctly, **B** finds the identity of corrupted wires (more than  $\frac{t_b}{2}$ ) after performing local comparison of  $P_j^{\mathbf{A}}$  and  $P_j^{\mathbf{B}}$ .

This completes the proof of the lemma. □

**Lemma 11.10** *The communication complexity of protocol Error-Identification-Static is  $\mathcal{O}(n^2 t_b)$ .*

*Proof:* During first phase, **A** sends an RS codeword of length  $n$  for  $n$  polynomials, thus communicating  $\mathcal{O}(n^2)$  field elements. During second phase, in the worst case, **B** broadcasts  $n$  error-lists, each containing at most  $\frac{t_b}{2}$  pairs, thus communicating  $\mathcal{O}(n^2 t_b)$  field elements. Communication complexity of conditional third phase is  $\mathcal{O}(n^2)$  field elements. Hence overall complexity is  $\mathcal{O}(n^2 t_b)$  field elements. □.

#### 11.4.2.1 Reducing the Communication Complexity of Error-Identification-Static

In protocol Error-Identification-Static, the most communication oriented step is in **Phase II**, where **B** may have to broadcast  $n$  error lists to reliably send them to **A**. This incurs a communication cost of  $\mathcal{O}(n^2 t_b)$ . We now present a nice trick to reduce the communication complexity of sending  $n$  error-lists from  $\mathcal{O}(n^2 t_b)$  to  $\mathcal{O}(n^2)$  during **Phase II** of protocol Error-Identification-Static, without changing the properties of the protocol.

Let  $Error\_List_J$  be the error-list with maximum number of pairs  $L_J$ , where  $J \in \{1, 2, \dots, n\}$ . If there are several error-lists with  $L_J$  pairs, then **B** arbitrarily selects one. **B** then broadcasts *only*  $Error\_List_J$  and sends the remaining error-lists concatenated into a list  $Y$ , by executing protocol 1-PRMT-Mixed( $Y, |Y|, n, t_b, t_f, L_j$ ) with **increased throughput**. **A** correctly receives  $Error\_List_J$  and verifies whether it is "good". If it is good, then **A** concludes that **B** has correctly recovered  $p_j^{\mathbf{A}}(x)$ . Moreover, **A** will identify  $L_J$  faulty wires from  $Error\_List_J$  as all the pairs listed in  $Error\_List_J$  are indeed corrupted values. Now from Theorem 5.7, protocol 1-PRMT-Mixed with increased throughput will correctly deliver the list  $Y$  containing the remaining error-lists. The rest of the protocol will now be same.

<sup>2</sup>This case is similar to Property 2.25 as explained in Chapter 2.

On the other hand, if  $\mathbf{A}$  finds that  $Error\_List_J$  is "bad", then  $\mathbf{A}$  concludes that  $\mathbf{B}$  has not recovered  $p_j^{\mathbf{A}}(x)$  correctly. In this case,  $\mathbf{A}$  fails to know  $L_J$  faults from  $Error\_List_J$  and hence 1-PRMT-Mixed will fail to deliver  $Y$  correctly. But  $\mathbf{A}$  identifies one polynomial, namely  $p_j^{\mathbf{A}}(x)$ , which is not recovered correctly by  $\mathbf{B}$  (due to more than  $\frac{t_b}{2}$  errors during **Phase I**). Note that while the properties of protocol *Error-Identification-Static* (Lemma 11.9) remain intact by incorporating these changes, the communication complexity reduces to  $\mathcal{O}(n^2)$ . We now provide the complete description of *Error-Identification-Static* that incorporates the above mentioned changes for attaining a communication complexity of  $\mathcal{O}(n^2)$  in Fig. 11.3.

We now prove the properties of protocol *Error-Identification-Static*.

**Lemma 11.11** *The communication complexity of new Error-Identification-Static (presented in Fig. 11.3) after incorporating new steps is  $\mathcal{O}(n^2)$ .*

*Proof:* During first phase,  $\mathbf{A}$  sends an RS codeword of length  $n$  for  $n$  polynomials, thus communicating  $\mathcal{O}(n^2)$  field elements. During second phase, broadcasting a *single* error-list ( $Error\_List_J$ ) requires communicating  $\mathcal{O}(n^2)$  field elements. From Theorem 5.7, sending the remaining error-lists by executing 1-PRMT-Mixed( $Y, |Y|, n, t_b, t_f, L_J$ ) with increased throughput will require communication of  $\mathcal{O}\left(\frac{|Y|}{L_J} * n\right) = \mathcal{O}(n^2)$  field elements as  $|Y| \leq (n-1) \times (2L_J)$ . Communication complexity of conditional third phase is  $\mathcal{O}(n^2)$  field elements. Hence overall complexity is  $\mathcal{O}(n^2)$  field elements.  $\square$

**Lemma 11.12** *The properties given in Lemma 11.9 will hold for new Error-Identification-Static (presented in Fig. 11.3) even after incorporating new steps.*

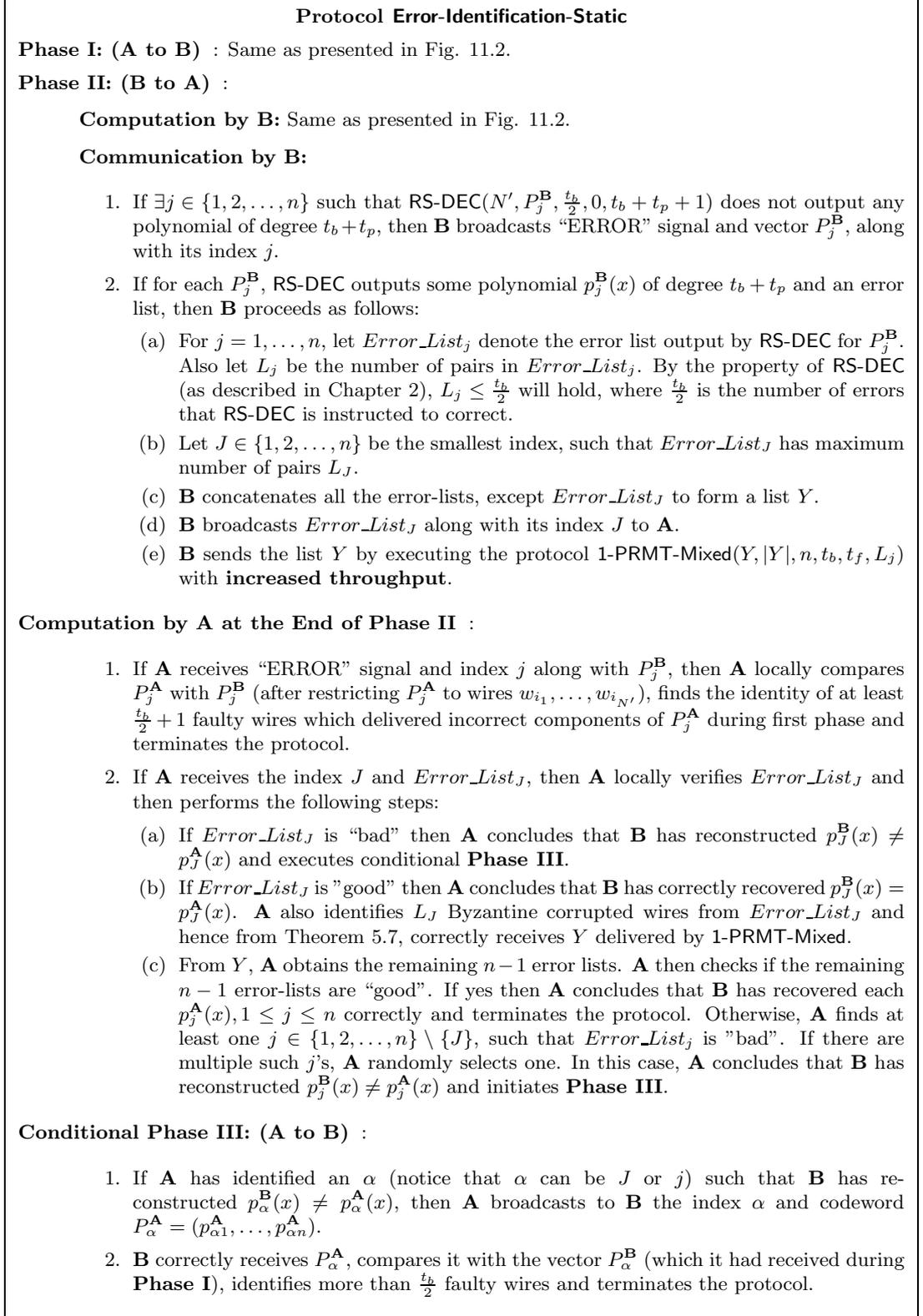
PROOF: If at most  $\frac{t_b}{2}$  Byzantine errors occur during **Phase I**, then each of the  $n$  error lists will be "good". In this case,  $\mathbf{A}$  on receiving  $Error\_List_J$  will identify that it is "good" and hence will know the identity of  $L_J$  Byzantine corrupted wires. So from Theorem 5.7, protocol 1-PRMT-Mixed with increased throughput will correctly deliver the remaining  $n-1$  error lists.  $\mathbf{A}$  will find all the remaining  $n-1$  error lists to be "good" and hence he concludes that  $\mathbf{B}$  has recovered each  $p_j^{\mathbf{A}}(x)$  correctly. Thus the vector  $p = (p_1^{\mathbf{A}}(0), \dots, p_n^{\mathbf{A}}(0))$  will be established correctly between  $\mathbf{A}$  and  $\mathbf{B}$ . The secrecy of  $p = (p_1^{\mathbf{A}}(0), \dots, p_n^{\mathbf{A}}(0))$  can be argued in the same way as done in Lemma 11.9.

Now consider the case when more than  $\frac{t_b}{2}$  wires are Byzantine corrupted during **Phase I**. Then, we have the following two cases:

1. If there exists some  $j \in \{1, 2, \dots, n\}$  such that after applying RS-DEC on  $P_j^{\mathbf{B}}$ ,  $\mathbf{B}$  does not obtain any  $t_b + t_p$  degree polynomial, then property **2** of Lemma 11.9 will follow from the proof of part (a) of **Case II** of Lemma 11.9.
2. If for each  $P_j^{\mathbf{B}}$ , RS-DEC outputs a polynomial of degree  $t_b + t_p$ , then we have the following two sub-cases:

- (a) **Sub-Case I:** If  $Error\_List_J$  is "good", then it implies that  $\mathbf{B}$  has correctly recovered  $p_j^{\mathbf{A}}(x)$ . In this case,  $\mathbf{A}$  on receiving  $Error\_List_J$  will also conclude the same. Moreover, since each value in  $Error\_List_J$  is indeed corrupted, the wires which delivered those values to  $\mathbf{B}$  during **Phase I** are Byzantine corrupted. Thus  $\mathbf{A}$  will know the identity of  $L_J$  Byzantine corrupted wires from  $Error\_List_J$ . Now, from Theorem 5.7, protocol 1-PRMT-Mixed with increased throughput will correctly deliver the remaining  $n-1$  error lists, concatenated to a list  $Y$ . Since more than  $\frac{t_b}{2}$  Byzantine errors have occurred during **Phase I**, at least one of the error lists in the remaining  $n-1$  error lists,

Figure 11.3: A Three Phase Protocol to Identify More than  $\frac{t_b}{2}$  Byzantine Faults with a Communication Complexity of  $\mathcal{O}(n^2)$



say  $Error\_List_j$ , will be "bad", which will be identified by **A**. In this case, **A** will execute the conditional **Phase III** and hence at the end of **Phase III**, **B** will know the identity of Byzantine corrupted wires (more than  $\frac{t_b}{2}$ ), which delivered incorrect components of  $P_j^A$  during **Phase I**. Thus property 2 of Lemma 11.9 will hold.

- (b) **Sub-Case II:** If  $Error\_List_j$  is "bad", then it implies that **B** has not correctly recovered  $p_j^A(x)$  because of more than  $\frac{t_b}{2}$  corrupted values in  $P_j^B$ . In this case, **A** on receiving  $Error\_List_j$  will conclude the same and will execute conditional **Phase III**. At the end of **Phase III**, **B** will know the identity of Byzantine corrupted wires (more than  $\frac{t_b}{2}$ ), which delivered incorrect components of  $P_j^A$  during **Phase I**. Thus property 2 of Lemma 11.9 will hold in this case as well.

This completes the proof of the lemma.  $\square$

### 11.4.3 Protocol 4-Optimal-PSMT-Static-Mixed

We now design a *four* phase communication optimal PSMT protocol called 4-Optimal-PSMT-Static-Mixed tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , where **S** and **R** are connected by  $n = 2t_b + t_f + t_p + 1$  wires. The protocol securely sends a message  $m^S$  containing  $n$  field elements by communicating  $\mathcal{O}(n^2)$  field elements. The protocol uses Error-Identification-Static and Pad-Establishment-Static as sub-protocols. By using these two sub-protocols, **S** and **R** tries to establish an information theoretically secure pad containing  $n$  field elements. Once this is done, **S** can mask the message by X-ORing it with the pad and broadcasts the masked message to **R**. On receiving the masked message, **R** unmaskes the message by X-ORing the masked message with the pad. The protocol is presented in Fig. 11.4.

We now prove the properties of protocol 4-Optimal-PSMT-Static-Mixed.

**Lemma 11.13 (Correctness)** *Protocol 4-Optimal-PSMT-Static-Mixed correctly delivers  $m^S$  in four phases.*

PROOF: In the protocol,  $m^S$  is masked by **S** using either the pad  $p$  or  $q$ . To prove the lemma, we show that **R** will also get the same pad in four phases. In the protocol, there are following two possibilities:

1. **Case I: Error-Identification-Static terminates in two phases:** Here there are further two possibilities:
  - (a) **Sub-Case (a): At the end of second phase, R concludes that the pad  $p$  is correctly established with S:** In this case, **R** broadcasts "SUCCESS-R" signal to **S** during **Phase III**. So at the end of **Phase III**, **S** will know that pad  $p$  is established between **S** and **R**.
  - (b) **Sub-Case (b): At the end of Phase II, R identifies at least  $\frac{t_b}{2} + 1$  Byzantine corrupted wires:** In this case, with the knowledge of  $\frac{t_b}{2} + 1$  Byzantine corrupted wires, **R** executes the single phase protocol Pad-Establishment-Static to establish the pad  $q$ . From Lemma 11.6, **S** will correctly get the pad  $q$  at the end of **Phase III**.
2. **Case II: Error-Identification-Static terminates in three phases:** In this case, **S** will identify at least  $\frac{t_b}{2} + 1$  Byzantine corrupted wires at the end of **Phase III** (Lemma 11.9). Now with the knowledge of  $\frac{t_b}{2} + 1$  Byzantine corrupted wires, **S**

Figure 11.4: A Four Phase Communication Optimal PSMT Protocol Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ ,  $n = 2t_b + t_f + t_p + 1$ ,  $|m^S| = n$

**Protocol 4-Optimal-PSMT-Static-Mixed-Protocol**

**R** and **S** starts executing protocol Error-Identification-Static, where **Phase I** is initiated by **R**.

1. If at the end of **Phase II** of protocol Error-Identification-Static, **R** finds that the pad  $p = (p_1^R(0), \dots, p_n^R(0))$  is established securely between **R** and **S**, then **R** broadcasts “SUCCESS-R” signal to **S** during **Phase III**. **S** on receiving the signal “SUCCESS-R”, computes  $\Gamma = m^S \oplus p$ , broadcasts  $\Gamma$  to **R** during **Phase IV** and terminates the protocol. **R** correctly receives  $\Gamma$ , recovers  $m^S = \Gamma \oplus p$  and terminates the protocol.
2. If at the end of **Phase II** of protocol Error-Identification-Static, **R** identifies at least  $\frac{t_b}{2} + 1$  Byzantine corrupted wires, then **R** securely establishes the one time pad  $q = (q_1^R(0), \dots, q_n^R(0))$  with **S** at the end of **Phase III** by executing the single phase protocol Pad-Establishment-Static. Once this is done, **S** computes  $\Gamma = m^S \oplus q$ , broadcasts  $\Gamma$  to **R** during **Phase IV** and terminates the protocol. **R** correctly receives  $\Gamma$ , recovers  $m^S = \Gamma \oplus q$  and terminates the protocol.
3. If conditional **Phase III** of protocol Error-Identification-Static is executed, then **S** identifies at least  $\frac{t_b}{2} + 1$  Byzantine corrupted wires at the end of third phase. **S** then securely establishes the one time pad  $q = (q_1^S(0), \dots, q_n^S(0))$  with **R** by executing the single phase protocol Pad-Establishment-Static during **Phase IV**. Moreover, **S** also computes  $\Gamma = m^S \oplus q$ , broadcasts  $\Gamma$  to **R** during **Phase IV** and terminates the protocol. **R** gets the pad  $q$  at the end of **Phase IV**, recovers  $m^S = \Gamma \oplus q$  and terminates the protocol.

executes the single phase protocol Pad-Establishment-Static to establish the pad  $q$ . From Lemma 11.6, **R** will correctly get the pad  $q$  at the end of **Phase IV**.

This completes the proof of the lemma.  $\square$

**Lemma 11.14 (Security)** *In protocol 4-Optimal-PSMT-Static-Mixed,  $m^S$  will be information theoretically secure.*

PROOF: The secrecy of  $m^S$  depends upon the secrecy of the pad, using which  $m^S$  is masked. If  $p$  is used as the masking pad, then secrecy of  $m^S$  follows from the secrecy of pad  $p$  (see Lemma 11.9). On the other hand, if  $q$  is used as the masking pad, then secrecy of  $m^S$  follows from the secrecy of pad  $q$  (see Lemma 11.7).  $\square$

**Lemma 11.15 (Communication Complexity)** *Communication complexity of protocol 4-Optimal-PSMT-Static-Mixed is  $\mathcal{O}(n^2)$ .*

PROOF: From Lemma 11.11 and Lemma 11.8, establishing the pad by executing protocols Error-Identification-Static and Pad-Establishment-Static incurs a communication cost of  $\mathcal{O}(n^2)$ . Moreover, since the message size is  $n$ , the masked message  $\Gamma$  will also contain  $n$  field elements and hence broadcasting it will require communicating  $\mathcal{O}(n^2)$  field elements. Thus the communication complexity of the protocol is  $\mathcal{O}(n^2)$ .  $\square$

**Theorem 11.16** *Protocol 4-Optimal-PSMT-Static-Mixed is an efficient, four phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .*

PROOF: The proof that 4-Optimal-PSMT-Static-Mixed is a four phase PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  follows from Lemma 11.13 and Lemma 11.14. From Theorem 11.5, any four phase PSMT protocol over  $n = 2t_b + t_f + t_p + 1$  wires must communicate  $\Omega(n^2)$  field elements to securely send a message containing  $n$  field elements against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . From Lemma 11.15, the total communication complexity of protocol 4-Optimal-PSMT-Static-Mixed is  $\mathcal{O}(n^2)$ . Hence protocol is a communication optimal protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . It is easy to see that both **S** and **R** performs polynomial computation in the protocol.  $\square$

Finally we state the following remark, which states how protocol 4-Optimal-PSMT-Static-Mixed overcomes the limitation of protocol 3-Optimal-PSMT-Static against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , as stated in Section 11.3.

**Remark 11.17** *Note that 4-Optimal-PSMT-Static-Mixed sends only codeword of polynomials, in contrast to the protocol 3-Optimal-PSMT-Static-Byzantine, which sends both polynomial and its codeword. The advantage that we get by sending only codeword is that we obtain one information theoretic secure value per codeword (after some intermediate information exchanges and then applying RS decoding). In the next chapter, we will show that this technique can be used to design communication optimal PSMT protocols even against mobile mixed adversary.*

## 11.5 Concluding Remarks and Open Problems

In this chapter, we presented a four phase communication optimal PSMT protocol in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . This, along with Theorem 11.4 and Theorem 11.5 completely settles the issue of POSSIBILITY, FEASIBILITY and OPTIMALITY of PSMT in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . These results are summarized in Fig. 11.5.

Figure 11.5: Summary of the Results for PSMT in Undirected Synchronous Network Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

Number of Phases ( $r$ )	Connectivity Requirement ( $n$ )	Lower Bound on Communication Complexity	Upper Bound
$r = 1$	$n \geq 3t_b + t_f + t_p + 1$ Theorem 11.1	$\Omega\left(\frac{n\ell}{n-(3t_b+t_f+t_p)}\right)$ Theorem 11.3	Theorem 11.3
$r \geq 2$	$n \geq 2t_b + t_f + t_p + 1$ Theorem 11.4	$\Omega\left(\frac{n\ell}{n-(2t_b+t_f+t_p)}\right)$ Theorem 11.5	Protocol 4-Optimal-PSMT-Static-Mixed: $n = 2t_b + t_f + t_p + 1, \ell = n$ Communication complexity = $\mathcal{O}(n\ell) = \mathcal{O}(n^2)$

From Fig. 11.5, we find that protocol 4-Optimal-PSMT-Static-Mixed is communication optimal only if the message contains  $\ell = n$  field elements. This leads to the following open problem:

**Open Problem 12** *Let **S** and **R** be connected by  $n = 2t_b + t_f + t_p + 1$  wires. Then does there exist a multiphase (two or more phase) PSMT protocol which securely sends a*

message containing  $\ell$  field elements by communicating  $\mathcal{O}(n\ell)$  field elements, tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , for any value of  $\ell$ ?

## Chapter 12

# PSMT in Undirected Networks Tolerating Mobile Mixed Adversary

In the previous chapter, we have seen that it is worth to study PSMT in the context of mixed adversary. Continuing this study further, in this chapter, we study PSMT in *undirected synchronous network, tolerating mobile mixed adversary*. The mobile mixed adversary, denoted by  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , may control different set of  $t_b, t_f$  and  $t_p$  wires in each phase of the protocol. Since  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  is more powerful than  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , the characterization and lower bound on communication complexity of single phase and multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  will also hold for  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Unfortunately, neither the three phase communication optimal PSMT protocol 3-Optimal-PSMT-Mobile-Byzantine tolerating  $\mathcal{A}_{t_b}^{mobile}$  (see Fig. 10.1), nor the four phase communication optimal PSMT protocol 4-Optimal-PSMT-Static-Mixed tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  (see Fig. 11.4) can be extended in a straight forward manner to design communication optimal PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .

The contribution of this chapter is two fold: we first design a nine phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + t_f + t_p + 1$  wires. The protocol uses certain ideas used in protocol 4-Optimal-PSMT-Static-Mixed. The protocol also uses protocol 3-Optimal-PRMT-Mobile-Mixed (see Fig. 6.2) as a black box.

After the publication of the above protocol in [19], Kurosawa et al. [42] presented a two phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + 1$  wires. To design their protocol, Kurosawa et al. introduced the concept of *pseudo-basis*, which was done for the first time in the literature of RMT/SMT. As a second contribution of this chapter, we design a three phase communication PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + t_f + t_p + 1$  wires. To design our three phase protocol, we extend the ideas used in the two phase protocol of [42] to the case of mixed adversary.

We would like to stress that the techniques used in our nine phase and three phase communication optimal PSMT protocol are completely independent and incomparable. While our nine phase protocol requires more phases than our three phase protocol, our nine phase protocol is communication optimal if  $\ell = \Theta(n)$ , while our three phase protocol is communication optimal if  $\ell = \Theta(n^2)$ . So we can decide which protocol to execute, depending upon whether reduced communication complexity is desirable or reduced phase complexity is desired.

We now discuss the network model and adversary settings used in this chapter.

## 12.1 Network Model and Adversary Settings

The network model and adversary settings used in this chapter is similar to the one used in Chapter 10. Thus we assume that there are  $n$  bi-directional, synchronous wires  $w_1, \dots, w_n$  between  $\mathbf{S}$  and  $\mathbf{R}$ . However, instead of  $\mathcal{A}_{t_b}^{mobile}$ , we assume the presence of a computationally unbounded mobile mixed adversary  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , who may corrupt *different* set of  $t_b, t_f$  and  $t_p$  wires in Byzantine, failstop and passive fashion respectively, during different phases of the protocol.

Though  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  controls different set of wires in different phases of the protocol, it does not allow the adversary to gain any information which has previously passed (in earlier phases of the protocol) through the wires under its control in current phase. The mobile mixed adversary gain information from the wires in a cumulative fashion. For example, suppose during first phase of a protocol,  $\mathcal{A}_{(1,1,1)}^{mobile}$  controls  $w_1, w_2$  and  $w_3$  in Byzantine, fail-stop and passive fashion respectively in a network, where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by wires  $w_1, w_2, \dots, w_5$ . Now suppose during second phase, it controls  $w_2, w_4$  and  $w_5$  in Byzantine, fail-stop and passive fashion respectively. Then  $w_1$  and  $w_3$  will behave correctly during second phase and adversary will have no access to the information passing through them in second phase. At the end of second phase, adversary will know the information which passed through  $w_1$  and  $w_3$  during first phase and the information that passed through  $w_2$  and  $w_5$  during second phase.

## 12.2 Characterization and Lower Bound on Communication Complexity for PSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$

Any single phase PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  will also work against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Thus Theorem 11.1 and Theorem 11.3 will also hold against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . This completely resolves the issue of POSSIBILITY and OPTIMALITY of single phase PSMT in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .

Since  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  is more powerful than  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , the necessary condition for multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , as given in Theorem 11.4 will also hold against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Similarly, the lower bound on communication complexity, as given in Theorem 11.5 will also hold against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . To show that the sufficiency condition for multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , as given in Theorem 11.4, will also hold against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , we will design two communication optimal PSMT protocols tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  in the subsequent sections, where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + t_f + t_p + 1$  wires .

Before proceeding further, we show that neither protocol 3-Optimal-PSMT-Mobile-Byzantine nor protocol 4-Optimal-PSMT-Static-Mixed, which are communication optimal PSMT protocols tolerating  $\mathcal{A}_{t_b}^{mobile}$  and  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  respectively, can be extended in a straight forward manner to design communication optimal PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .

## 12.3 Limitations of Protocol 3-Optimal-PSMT-Mobile-Byzantine and 4-Optimal-PSMT-Static-Mixed Against $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$

The reason why our three phase communication optimal PSMT protocol 3-Optimal-PSMT-Mobile-Byzantine tolerating  $\mathcal{A}_{t_b}^{mobile}$  cannot be extended to design communication

optimal PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  is similar as given in Section 11.3. That is, if we extend protocol 3-Optimal-PSMT-Mobile-Byzantine against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , then **S** and **R** will end up sharing an information theoretically secure one time pad of size  $n(t_b + 1)$  by communicating  $\mathcal{O}(n^3)$  field elements. If  $n = 2t_b + t_f + t_p + 1$  then  $t_b$  may not be  $\Theta(n)$  and can even be a constant. Thus the resulting PSMT protocol may send a message of very small size with very high communication complexity of  $\mathcal{O}(n^3)$ , which will not be a communication optimal PSMT protocol against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .

The techniques used for designing protocol 4-Optimal-PSMT-Static-Mixed tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  cannot be reused for designing communication optimal PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . This is due to the failure of protocol Error-Identification-Static to achieve its properties with a communication of  $\mathcal{O}(n^2)$  in the presence of  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Recall that in Error-Identification-Static, **B** reliably sends  $n$  error lists during second phase. This was done by broadcasting the error list with maximum number of pairs  $L_J$  and then jointly sending the remaining  $n - 1$  error lists by executing protocol 1-PRMT-Mixed, with a block size of  $L_J$ . Also recall that when the maximum sized error list is "good", then from the error list, **A** will know the identities of  $L_J$  wires, which were Byzantine corrupted during first phase. Now since the adversary was static, *the same set of wires will be Byzantine corrupted in the second phase as well and hence A could neglect them.* This facilitated **A** to correctly recover the remaining  $n - 1$  error lists at the end of 1-PRMT-Mixed (see Theorem 5.7). However, this technique not work against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . This is because *the  $L_J$  wires which were Byzantine corrupted during first phase, may not be under the control of  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  in the second phase.* In the worst case, all these  $L_J$  wires may be completely honest and hence by neglecting them, **A** will loose information sent through  $L_J$  honest wires. Thus protocol 1-PRMT-Mixed with **increased throughput** will fail to correctly deliver the remaining  $n - 1$  error lists to **A** in the presence  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . So even when the maximum sized error list is "good", **A** will fail to reliably receive the remaining  $n - 1$  error lists.

To reliably send the error lists in the presence  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , **B** may broadcast all of them to **A**. Though broadcasting the  $n$  error lists ensures their proper delivery, it will increase the communication complexity of Error-Identification-Static from  $\mathcal{O}(n^2)$  to  $\mathcal{O}(n^2 t_b)$ . Thus the resultant PSMT protocol incorporating Error-Identification-Static will not be a communication optimal PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . In order to deal with this problem, we use our three phase communication optimal PRMT protocol 3-Optimal-PRMT-Mobile-Mixed (see Fig. 6.2) tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ . Using this protocol as a black box, we design our nine phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , which is given in the next section.

## 12.4 Nine Phase Communication Optimal PSMT

Let **S** and **R** be connected by  $n = 2t_b + t_f + t_p + 1$  wires. We now present a *nine* phase communication optimal PSMT protocol called 9-Optimal-PSMT-Mobile-Mixed, which securely sends a message containing  $\Theta(n)$  field elements by communicating  $\mathcal{O}(n^2)$  field elements against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . The protocol uses few ideas from protocol 4-Optimal-PSMT-Static-Mixed and uses protocol 3-Optimal-PRMT-Mobile-Mixed as a black-box. The protocol establishes an information theoretically secure one time pad of length either  $n - 1 = \Theta(n)$  or  $\frac{n}{2} = \Theta(n)$  between **S** and **R**, depending upon the behavior of the adversary, by communicating  $\mathcal{O}(n^2)$  field elements. Accordingly, **S** sends a message containing either  $n - 1$  or  $\frac{n}{2}$  field elements by communicating  $\mathcal{O}(n^2)$  field elements.

Note that according to the description provided in Fig. 6.2, protocol 3-Optimal-PRMT-Mobile-Mixed is executed with  $2t_b + t_f + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ , tolerating  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ . So in protocol 9-Optimal-PSMT-Mobile-Mixed, protocol 3-Optimal-PRMT-Mobile-Mixed is executed with any set of predefined  $2t_b + t_f + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ . Thus  $\mathbf{S}$  and  $\mathbf{R}$  neglects a pre-determined set of  $t_p$  wires and runs protocol 3-Optimal-PRMT-Mobile-Mixed on the remaining  $2t_b + t_f + 1$  wires. This does not affect the correctness and working of protocol 9-Optimal-PSMT-Mobile-Mixed.

Informally, the idea of protocol 9-Optimal-PSMT-Mobile-Mixed is as follows:  $\mathbf{S}$  sends  $n$  random codewords to  $\mathbf{R}$ , where each codeword is RS encoded using a random polynomial of degree  $t_b + t_p$ .  $\mathbf{R}$  assumes that at most  $\frac{t_b}{2}$  values are corrupted in each received vector and tries to correct them. However,  $\mathbf{R}$  will not know the status of the recovered polynomials. So  $\mathbf{R}$  reliably sends back only the first  $\frac{n}{2}$  error lists.  $\mathbf{S}$  on receiving these lists can find out the status of these lists. If all the error lists are good, then  $\mathbf{S}$  knows that the first  $\frac{n}{2}$  polynomials are correctly recovered by  $\mathbf{R}$  and hence the constant term of these polynomials acts as a pad of size  $\frac{n}{2}$ .

On the other hand, if  $\mathbf{S}$  finds any of the first  $\frac{n}{2}$  error lists to be bad, then by interacting with  $\mathbf{R}$ , sender  $\mathbf{S}$  finds out the identity of more than  $\frac{t_b}{2} + 1$  wires, which were Byzantine corrupted during first phase.  $\mathbf{S}$  then again reliably resends the components of the last  $\frac{n}{2}$  codewords which were transmitted over those corrupted wires. Note that these values were already known to the adversary during the first phase and hence does not provide any extra information about the last  $\frac{n}{2}$  polynomials to the adversary. However,  $\mathbf{R}$  will now have sufficient correct values corresponding to the last  $\frac{n}{2}$  polynomials and hence  $\mathbf{R}$  can correctly recover the last  $\frac{n}{2}$  polynomials and the constant term of these polynomials acts as a pad of size  $\frac{n}{2}$ .

Protocol 9-Optimal-PSMT-Mobile-Mixed is given in Fig. 12.1, Fig. 12.2 and Fig. 12.3.

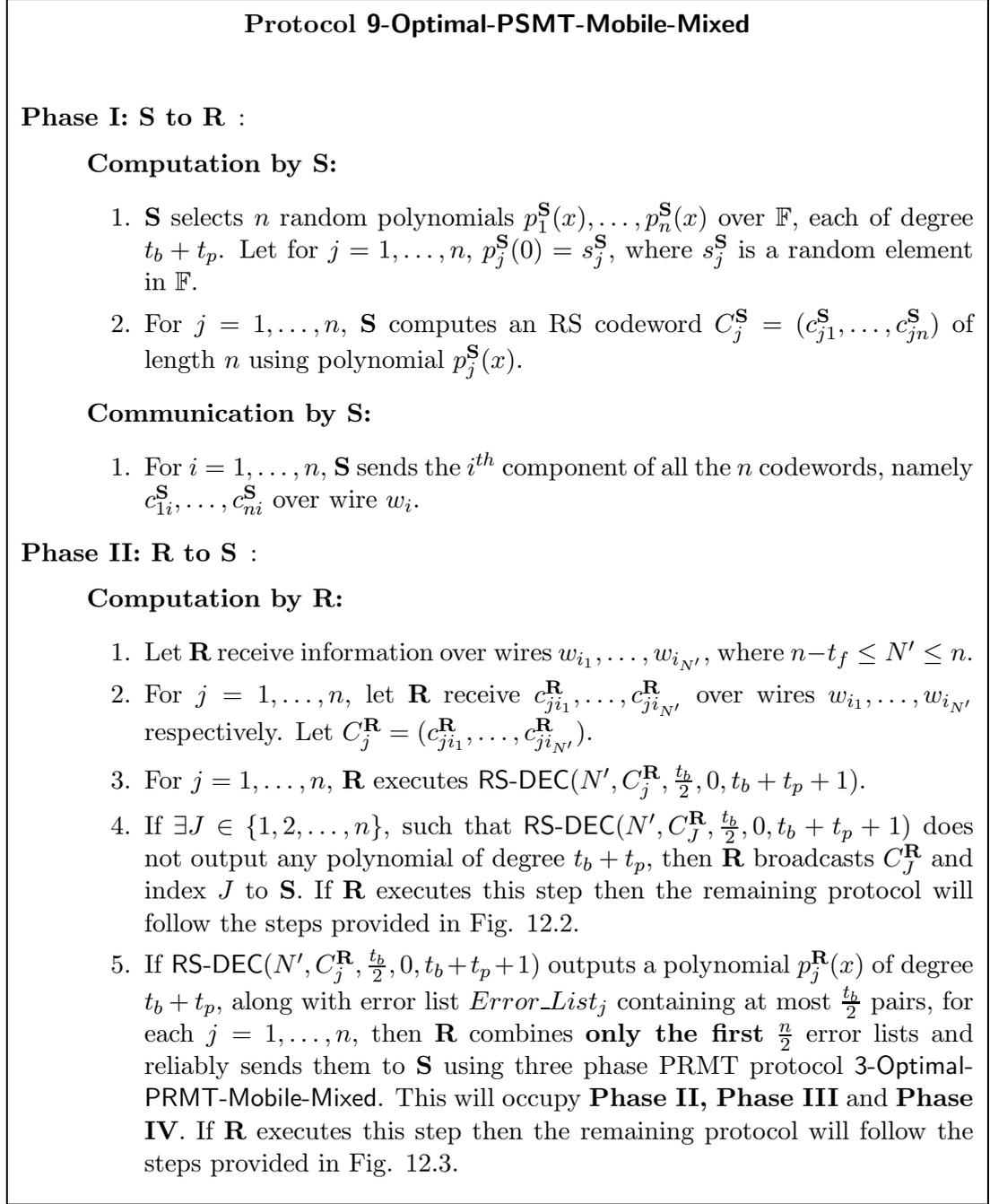
We now formally prove the properties of protocol 9-Optimal-PSMT-Mobile-Mixed.

**Theorem 12.1** *Protocol 9-Optimal-PSMT-Mobile-Mixed correctly and securely sends a message containing  $\Theta(n)$  field elements in at most nine phases by communicating  $\mathcal{O}(n^2)$  field elements tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .*

*Proof.* We prove the theorem for the worst case where exactly  $t_f$  wires (probably different set) fails to deliver any information in each phase due to fail-stop corruption. Thus each vector  $C_j^{\mathbf{R}}$  received during first phase will be of length  $N' = n - t_f = 2t_b + t_p + 1$ . Each  $C_j^{\mathbf{R}}$  corresponds to an RS codeword  $C_j^{\mathbf{S}}$ , encoded using a polynomial of degree  $t_b + t_p$ . Now consider the following two cases:

1. **Case I: At most  $\frac{t_b}{2}$  wires are Byzantine Corrupted During Phase I:** In this case, for each  $C_j^{\mathbf{R}}$ , RS-DEC( $N', C_j^{\mathbf{R}}, \frac{t_b}{2}, 0, t_b + t_p + 1$ ) will output the correct  $p_j^{\mathbf{S}}(x)$  and a corresponding "good" error list at the end of **Phase II**. But as RS-DEC does not has extra error detecting capability apart from the capability of correcting  $\frac{t_b}{2}$  errors,  $\mathbf{R}$  will not know whether reconstructed  $p_j^{\mathbf{S}}(x)$ 's are correct or not. So  $\mathbf{R}$  follows step 5 in **Phase II** and sends the first  $\frac{n}{2}$  error lists to  $\mathbf{S}$  by executing protocol 3-Optimal-PRMT-Mobile-Mixed. By the correctness of 3-Optimal-PRMT-Mobile-Mixed,  $\mathbf{S}$  will correctly receive all the  $\frac{n}{2}$  error lists and will find all of them to be "good". So  $\mathbf{S}$  will conclude that first  $\frac{n}{2}$  polynomials, namely  $p_1^{\mathbf{S}}(x), \dots, p_{\frac{n}{2}}^{\mathbf{S}}(x)$ , are recovered correctly by  $\mathbf{R}$ . Hence  $\mathbf{S}$  uses  $p = (p_1^{\mathbf{S}}(0), \dots, p_{\frac{n}{2}}^{\mathbf{S}}(0))$  as a pad to blind a message  $m^{\mathbf{S}}$  of size  $\frac{n}{2}$  and sends the blinded message  $\Gamma$  to  $\mathbf{R}$  by broadcasting it. Once  $\mathbf{R}$  receives  $\Gamma$ , he can recover the

Figure 12.1: A Nine Phase Communication Optimal PSMT Protocol Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}, n = 2t_b + t_f + t_p + 1$



message using  $\Gamma$  and the pad  $p = (p_1^{\mathbf{S}}(0), \dots, p_{\frac{n}{2}}^{\mathbf{S}}(0))$ . Thus in this case protocol 9-Optimal-PSMT-Mobile-Mixed sends  $\frac{n}{2}$  field elements in five phases.

2. **Case II: More than  $\frac{t_b}{2}$  wires are Byzantine Corrupted During Phase I:** This case may lead to further two subcases:

- (a) RS-DEC outputs some polynomial  $p_j^{\mathbf{R}}(x)$  of degree  $t_b + t_p$  for each  $C_j^{\mathbf{R}}$ ;
- (b) There exists a  $J \in \{1, 2, \dots, n\}$  for which RS-DEC fails to output any polynomial.

Figure 12.2: Remaining Execution of Protocol 9-Optimal-PSMT-Mobile-Mixed if **R** Executes Step 4 During **Phase II**. In this case,  $|m^{\mathbf{S}}| = n - 1$

**Execution I - Protocol 9-Optimal-PSMT-Mobile-Mixed Continued ...**  
**(This Part Will be Executed if R Executes Step 4 During Phase II)**

**Phase III: S to R :**

**Computation by S:**

1. **S** correctly receives index  $J$  and vector  $C_J^{\mathbf{R}}$ .
2. After locally comparing  $C_J^{\mathbf{R}}$  with the corresponding original codeword  $C_J^{\mathbf{S}}$ , **S** identifies at least  $\frac{t_b}{2} + 1$  wires which delivered incorrect components of  $C_J^{\mathbf{S}}$  to **R** during **Phase I**. **S** saves the identity of these wires in a list  $L_{fault}$ .
3. For  $j \in \{1, 2, \dots, n\} \setminus \{J\}$ , **S** lists all  $c_{ji}^{\mathbf{S}}$ 's, which were sent during **Phase I** over  $w_i \in L_{fault}$ , in a list **ReSendValues**. Thus  $|\mathbf{ReSendValues}| = (n - 1)|L_{fault}| \geq (n - 1)(\frac{t_b}{2} + 1)$ .
4. **S** constructs a pad  $p$  consisting of  $n - 1$   $s_j^{\mathbf{S}}$ 's with  $j \in \{1, 2, \dots, n\} \setminus \{J\}$  where  $s_j^{\mathbf{S}} = p_j^{\mathbf{S}}(0)$ .
5. With a message  $m^{\mathbf{S}}$  containing  $n - 1$  field elements, **S** computes  $\Gamma = m^{\mathbf{S}} \oplus p$ .

**Communication by S:**

1. **S** reliably sends  $L_{fault}$  and  $\Gamma$  to **R** by broadcasting it.
2. **S** reliably re-sends the values in list **ReSendValues** to **R** by executing the *three phase* PRMT protocol 3-Optimal-PRMT-Mobile-Mixed. This will run in **Phase III**, **Phase IV** and **Phase V**. **S** then terminates 9-Optimal-PSMT-Mobile-Mixed.

**Message Recovery by R (At the end of Phase V):**

1. **R** correctly receives  $\Gamma, L_{fault}$  and the values in **ReSendValues**.
2. From  $L_{fault}$ , **R** identifies  $|L_{fault}| > \frac{t_b}{2}$  wires which had delivered incorrect values during **Phase I**.
3. For  $j \in \{1, 2, \dots, n\} \setminus \{J\}$ , corresponding to each  $C_j^{\mathbf{R}}$ , **R** replaces the  $c_{ji}^{\mathbf{R}}$ 's, which were received during **Phase I** over  $w_i \in L_{fault}$ , with the corresponding actual  $c_{ji}^{\mathbf{S}}$ 's from the list **ReSendValues**.
4. After replacement, **R** knows that for each  $j \in \{1, 2, \dots, n\} - \{J\}$ , at most  $t_b - |L_{fault}|$  values could be corrupted in  $C_j^{\mathbf{R}}$ . **R** executes  $\text{RS-DEC}(N', C_j^{\mathbf{R}}, t_b - |L_{fault}|, 0, t_b + t_p + 1)$  to recover  $p_j^{\mathbf{S}}(x)$ .
5. Once the  $p_j^{\mathbf{S}}(x)$ 's for  $j \in \{1, 2, \dots, n\} \setminus \{J\}$  are obtained, **R** constructs the pad  $p$  in the same way as done by **S**.
6. **R** computes  $m^{\mathbf{S}} = \Gamma \oplus p$  and terminates protocol 9-Optimal-PSMT-Mobile-Mixed.

Figure 12.3: Remaining Execution of Protocol 9-Optimal-PSMT-Mobile-Mixed if **R** Executes Step 5 During **Phase II**. In this Case,  $|m^S| = \frac{n}{2}$ .

**Execution II - Protocol 9-Optimal-PSMT-Mobile-Mixed Continued ...**  
**(This Part Will be Executed if R Executes Step 5 During Phase II)**

**Local Computation by S (At the end of Phase IV):**

1. **S** correctly receives the first  $\frac{n}{2}$  error lists, sent by **R** using protocol 3-Optimal-PRMT-Mobile-Mixed.
2. **S** then checks the status of these error lists.
  - (a) If all the  $\frac{n}{2}$  error lists are "good", then **S** concludes that **R** has correctly recovered  $p_j^S(x)$  for  $j = 1, \dots, \frac{n}{2}$  and an information theoretically secure pad  $p = (p_1^S(0), \dots, p_{\frac{n}{2}}^S(0))$  is established with **R**. So, **S** considers a message  $m^S$  containing  $\frac{n}{2}$  field elements and computes  $\Gamma = m^S \oplus p$ .
  - (b) Else  $\exists J \in \{1, 2, \dots, \frac{n}{2}\}$ , such that  $Error\_List_J$  is "bad". In this case, **S** concludes that more than  $\frac{t_b}{2}$  values have been changed in  $J^{th}$  codeword during **Phase I**.

**Phase V: S to R:**

1. If **S** has computed  $\Gamma$ , then **S** broadcasts  $\Gamma$  and a terminating signal to **R**.
2. Else **S** broadcasts index  $J$  along with "ERROR" signal, asking **R** to broadcast the  $J^{th}$  vector  $C_J^R$  as received by **R** during **Phase I**.

**Local Computation by R (At the End of Phase V):**

1. If **R** receives terminating signal and  $\Gamma$ , then **R** concludes that it has correctly recovered  $p_1^S(x), \dots, p_{\frac{n}{2}}^S(x)$  during **Phase I**. **R** then forms the pad  $p = (p_1^S(0), \dots, p_{\frac{n}{2}}^S(0))$ , computes  $m^S = p \oplus \Gamma$  and terminates protocol 9-Optimal-PSMT-Mobile-Mixed.
2. Else **R** receives index  $J$  and "ERROR" signal. In this case, **R** concludes that more than  $\frac{t_b}{2}$  corrupted values are present in  $C_J^R$ . So **R** executes **Phase VI** as follows:

**Phase VI: R to S:**

1. **R** broadcasts the vector  $C_J^R$  to **S**.

**Local Computation by S (At the End of Phase VI):**

1. Upon receiving  $C_J^R$  and comparing it with  $C_J^S$ , **S** identifies more than  $\frac{t_b}{2}$  wires which were Byzantine corrupted during **Phase I** and saves them in a list  $L_{fault}$ .
2. Corresponding to each  $j \in \{\frac{n}{2} + 1, \dots, n\}$  and each  $w_i \in L_{fault}$ , **S** adds the value  $c_{j_i}^S$  to a list **ReSendValues**.
3. With the pad  $p = (p_{\frac{n}{2}+1}^S(0), \dots, p_n^S(0))$  and a message  $m^S$  containing  $\frac{n}{2}$  field elements, **S** computes  $\Gamma = m^S \oplus p$ .

**Phase VII: S to R:**

1. **S** reliably sends  $\Gamma$  and  $L_{fault}$  to **R** by broadcasting them.
2. **S** reliably sends **ReSendValues** by executing the *three* phase PRMT protocol 3-Optimal-PRMT-Mobile-Mixed and terminates protocol 9-Optimal-PSMT-Mobile-Mixed. Protocol 3-Optimal-PRMT-Mobile-Mixed will run in **Phase VII**, **Phase VIII** and **Phase IX**.

**Message Recovery by R at the End of Phase IX:**

1. **R** recovers  $m^S$  in the same way as in **Execution I**, given in Fig. 12.2. The only difference is that **R** performs the computation with respect to vectors  $C_{\frac{n}{2}+1}^R, \dots, C_n^R$ .

While in first subcase, occurrence of more than  $\frac{t_b}{2}$  faults cannot be immediately detected by **R** (as RS-DEC is applied with  $d = 0$ ), in the second subcase, it can be immediately detected by **R**.

Now in the first subcase, if more than  $\frac{t_b}{2}$  Byzantine errors occur in the codewords of *only* last  $\frac{n}{2}$  polynomials, i.e, for  $p_j^S(x)$ , such that  $j = \frac{n}{2} + 1, \dots, n$  (this implies

that at most  $\frac{t_b}{2}$  Byzantine errors took place in the first  $\frac{n}{2}$  codewords), then the proof is same as in **Case I**. On the other hand, if more than  $\frac{t_b}{2}$  faults occurs for  $J^{th}$  codeword, where  $J \in \{1, 2, \dots, \frac{n}{2}\}$ , then the proof proceeds as follows:

- **During Phase II,  $\mathbf{R}$  reconstructs  $p_J^{\mathbf{R}}(x) \neq p_J^{\mathbf{S}}(x)$ ,  $J \in \{1, 2, \dots, \frac{n}{2}\}$ :** In this case,  $Error\_List_J$  is a “bad” error list. Since  $\mathbf{R}$  reliably sends back first  $\frac{n}{2}$  error lists using 3-Optimal-PRMT-Mobile-Mixed,  $\mathbf{S}$  correctly receives  $Error\_List_J$  and finds it to be ”bad”.  $\mathbf{S}$  concludes that  $\mathbf{R}$  has reconstructed some  $p_J^{\mathbf{R}}(x) \neq p_J^{\mathbf{S}}(x)$ . So  $\mathbf{S}$  asks  $\mathbf{R}$  to broadcast the  $J^{th}$  vector  $C_J^{\mathbf{R}}$ , as received during **Phase I**. On receiving  $C_J^{\mathbf{R}}$ ,  $\mathbf{S}$  compares it with its corresponding original codeword  $C_J^{\mathbf{S}}$  and identifies  $|L_{fault}| \geq \frac{t_b}{2} + 1$  wires which delivered incorrect values to  $\mathbf{R}$  during **Phase I**. Now by executing 3-Optimal-PRMT-Mobile-Mixed,  $\mathbf{S}$  re-sends the components of the last  $\frac{n}{2}$  codewords, which were sent through these corrupted wires during **Phase I**.  $\mathbf{S}$  also broadcasts the identity of these corrupted wires. *Note that re-sending these values, does not leak any additional information about the last  $\frac{n}{2}$   $p_j^{\mathbf{S}}(x)$ 's to  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  as the adversary already knew these values during **Phase I**.* But now with the new values received,  $\mathbf{R}$  have  $N' = 2t_b + t_p + 1$  components for each of the last  $\frac{n}{2}$  vectors and at most  $t_b - |L_{fault}| \leq \frac{t_b}{2} - 1$  of these  $N'$  components could be corrupted. By substituting  $N' = 2t_b + t_p + 1, c = t_b - |L_{fault}|, d = 0$  and  $k = t_b + t_p + 1$  in the inequality of Theorem 2.19, we find that RS-DEC( $N', C_J^{\mathbf{R}}, t_b - |L_{fault}|, 0, t_b + t_p + 1$ ), can correct all the remaining  $t_b - |L_{fault}| < \frac{t_b}{2}$  errors present in  $C_J^{\mathbf{R}}$  and can output the corresponding polynomial  $p_j^{\mathbf{S}}(x)$  where  $j \in \{\frac{n}{2} + 1, \dots, n\}$ .  $\mathbf{R}$  then considers the constant term of these last  $\frac{n}{2}$   $p_j^{\mathbf{S}}(x)$ 's as the secret pad established with  $\mathbf{S}$ . The secrecy of the pad follows from the fact that at any stage of the execution,  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  will not get more than  $t_b + t_p$  points on the last  $\frac{n}{2}$   $p_j^{\mathbf{S}}(x)$ 's, each of which is of degree  $t_b + t_p$ . Once  $\mathbf{R}$  obtains the pad, he can compute the message from the blinded message  $\Gamma$ . Thus in this case protocol 9-Optimal-PSMT-Mobile-Mixed sends  $\frac{n}{2}$  field elements in nine phases.
- **During Phase II,  $\mathbf{R}$  is Unable to Recover Some  $p_J^{\mathbf{R}}(x)$ , Where  $J \in \{1, \dots, \frac{n}{2}\}$ :** In this case  $\mathbf{R}$  broadcasts only the  $J^{th}$  received codeword  $C_J^{\mathbf{R}}$ , from which  $\mathbf{S}$  (after local verification) identifies at least  $\frac{t_b}{2} + 1$  wires, which delivered incorrect values to  $\mathbf{R}$  during **Phase I**. Now the rest of the proof is same as in the above case. The only difference is that here a pad of length  $n - 1$  will be established between  $\mathbf{S}$  and  $\mathbf{R}$ .

In protocol 9-Optimal-PSMT-Mobile-Mixed,  $\mathbf{S}$  communicates  $\mathcal{O}(n^2)$  field elements for sending  $n$  codewords during the first phase. In second phase  $\mathbf{R}$  either sends a codeword or  $\frac{n}{2}$  error lists each of size at most  $\frac{t_b}{2}$ . Sending the codeword by broadcasting requires  $\mathcal{O}(n^2)$  communication complexity. On the other hand, sending  $\frac{n}{2}$  error lists each of size at most  $\frac{t_b}{2}$  (so total  $\frac{n}{2} * \frac{t_b}{2} = \Theta(nt_b)$  field elements) using 3-Optimal-PRMT-Mobile-Mixed also requires a communication of  $\mathcal{O}(n^2)$  field elements (see Lemma 6.14). Similarly, re-sending  $\Theta(nt_b)$  values corresponding to the codewords by executing 3-Optimal-PRMT-Mobile-Mixed requires communicating  $\mathcal{O}(n^2)$  field elements. It is easy to see that no more than  $\mathcal{O}(n^2)$  field elements are communicated in any other phase as well. Hence the overall communication complexity of the protocol is  $\mathcal{O}(n^2)$ .  $\square$

**Theorem 12.2** *Protocol 9-Optimal-PSMT-Mobile-Mixed is a nine phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .*

PROOF: As mentioned earlier, Theorem 11.5 will also hold against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . This implies that any nine phase PSMT over  $n = 2t_b + t_f + t_p + 1$  wires must communicate  $\Omega(n^2)$  field elements to securely send a message containing  $\Theta(n)$  field elements tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Since the total communication complexity of protocol 9-Optimal-PSMT-Mobile-Mixed is  $\mathcal{O}(n^2)$ , protocol 9-Optimal-PSMT-Mobile-Mixed is a communication optimal PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .  $\square$

## 12.5 Three Phase Communication Optimal PSMT Protocol Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$

After the publication of protocol 9-Optimal-PSMT-Mobile-Mixed in [19], Kurosawa et al. [42] presented a two phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , where **S** and **R** are connected by  $n = 2t_b + 1$  wires. To design their protocol, Kurosawa et al. introduced the concept of *pseudo-basis*, which was done for the first time in the literature of RMT/SMT. By extending this notion to the case of mixed adversary, we now design a three phase communication PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , where **S** and **R** are connected by  $n = 2t_b + t_f + t_p + 1$  wires.

### 12.5.1 Pseudo-Basis and Pseudo-Dimension

We now describe the notion of Pseudo-Basis and Pseudo-Dimension, a novel and interesting concept, introduced by Kurosawa et.al. in [42]. These notions were introduced with respect to  $\mathcal{A}_{t_b}^{static}$  to design a two phase polynomial time communication optimal PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ . We now extend these notions for mixed adversary and using them, we design our three phase communication optimal PSMT protocol called 3-Optimal-PSMT-Mobile-Mixed, tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .

**Assumption 12.3** *Without loss of generality, for the ease of exposition, we use the following assumption in the following discussion and also in protocol 3-Optimal-PSMT-Mobile-Mixed: if **S** (**R**) is expecting some information in some specific format from **R** (**S**) along a wire in a phase and if no/syntactically incorrect information comes, then **S** (**R**) substitutes some default value(s) from  $\mathbb{F}$  in the desired format and proceeds with his computation. Thus we separately do not consider the case when no/syntactically incorrect information is received along a wire.*

Now let **A** and **B** be two specific nodes (**A** (**B**) can be **S** or **R**) which are connected by  $n = 2t_b + t_f + t_p + 1$  wires denoted by  $w_1, \dots, w_n$ , of which at most  $t_b$ ,  $t_f$  and  $t_p$  wires can be corrupted in Byzantine, fail-stop and passive fashion respectively. Let  $\mathcal{C}$  be the set of all possible RS codewords of length  $n$  over  $\mathbb{F}$ , which are RS encoded using polynomials of degree  $t_b + t_p$  over  $\mathbb{F}$ . Moreover, let  $\alpha_1, \dots, \alpha_n$  be the publicly known elements from  $\mathbb{F}$ , at which the polynomials are evaluated to compute the RS codewords in  $\mathcal{C}$ . It is easy to see that the hamming distance [45] of code  $\mathcal{C}$  is  $n - (t_b + t_p) = t_b + t_f + 1$ . We may call the individual codewords in  $\mathcal{C}$  as  $n$ -dimensional vectors. But it should be noted that any codeword of length  $n$  is a vector of length  $n$ , but the reverse is not true.

If **A** sends several codewords, say  $\gamma$  codewords  $C_1, \dots, C_\gamma \in \mathcal{C}$  over the  $n$  wires by transmitting  $i^{th}$  component of all the codewords over  $i^{th}$  wire  $w_i$ , then the locations at which Byzantine and fail-stop errors occur in these codewords are not random. This is because for all the codewords, the Byzantine and fail-stop errors always occur at the same  $t_b + t_f$  (or a smaller subset) locations. The concept of pseudo-basis is based on this simple and interesting observation.

Let  $\mathbf{B}$  receive vectors  $Y_1 \dots, Y_\gamma$  over the  $n$  wires, such that for  $i = 1, \dots, \gamma$ ,

$$Y_i = C_i + E_i, \quad (12.1)$$

where

$$E_i = (e_{i1}, \dots, e_{in})$$

is an error vector introduced by the adversary. Since at most  $t_b$  Byzantine and at most  $t_f$  fail-stop corruptions could occur, each  $E_i$  may have at most  $t_b + t_f$  non-zero components. Let

$$\text{support}(E_i) = \{j \mid e_{ij} \neq 0\}. \quad (12.2)$$

Then there exists a set  $\{w_{j_1}, \dots, w_{j_{(t_b+t_f)}}\}$  of wires that are Byzantine and fail-stop corrupted, such that each error vector  $E_i$  satisfies

$$\text{support}(E_i) \subseteq \{j_1, \dots, j_{(t_b+t_f)}\} \quad (12.3)$$

This means that the space  $E$  spanned by the error vectors  $E_1, \dots, E_\gamma$  has dimension at most  $t_b + t_f$ . The notion of pseudo-basis exploits this idea.

Let  $\mathcal{V}$  denote the  $n$ -dimensional vector space over  $\mathbb{F}$ . We say that  $\{E_1, \dots, E_\gamma\}$  is the *real error vector set* of  $\mathcal{Y} = \{Y_1, \dots, Y_\gamma\}$ . We also say that  $E$  is the *real error vector space* if it is spanned by the real error vector set  $\{E_1, \dots, E_\gamma\}$ .

For two vectors  $V_1, V_2 \in \mathcal{V}$ , we write

$$V_1 = V_2 \text{ mod } \mathcal{C} \quad (12.4)$$

if  $V_1 - V_2 \in \mathcal{C}$ . Notice that for  $i = 1, \dots, \gamma$ , for every triplet  $(Y_i, C_i, E_i)$ ,

$$Y_i = E_i \text{ mod } \mathcal{C} \quad (12.5)$$

holds, as  $Y_i - E_i = C_i \in \mathcal{C}$ . We say that  $\{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$  is an *admissible error vector set* of  $\mathcal{Y}$  if each  $\mathcal{E}_i$  satisfies  $Y_i = C_i + \mathcal{E}_i$ , for some codeword  $C_i \in \mathcal{C}$  and

$$|\text{support}(\mathcal{E}_1) \cup \dots \cup \text{support}(\mathcal{E}_\gamma)| \leq t_b + t_f \quad (12.6)$$

We say that  $\mathcal{E}$  is an *admissible error vector space* of  $\mathcal{Y}$  if it is spanned by an admissible error vector set  $\{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$ . Notice that for a given  $\mathcal{Y}$ , there exists a unique real error vector set and real error vector space, while there may exist several admissible error vector set and corresponding admissible error vector space. Also notice that the real error vector set (real error vector space) is also an admissible error vector set (admissible error vector space) but the reverse may not be true. Even though an admissible error vector set  $\{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$  for a given  $\mathcal{Y}$  may not be unique, the results presented in the sequel hold for any admissible error vector set.

We begin with the definition of *linearly pseudo-express*.

**Definition 12.4 (Linearly Pseudo-Express [42])** *We say that a vector  $Y \in \mathcal{Y}$  is linearly pseudo-expressed by  $\{B_1, \dots, B_k\}$  if there exists some  $\alpha = (a_1, \dots, a_k)$ , such that*

$$Y = a_1 B_1 + \dots + a_k B_k \text{ mod } \mathcal{C}$$

We now state the following lemma:

**Lemma 12.5** *Let  $\{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$  be an admissible error vector set of  $\mathcal{Y}$ . Then  $\mathcal{E}_i$  is linearly expressed by  $\{\mathcal{E}_{j_1}, \dots, \mathcal{E}_{j_k}\}$  iff  $Y_i$  is linearly pseudo-expressed by  $\{Y_{j_1}, \dots, Y_{j_k}\}$ .*

PROOF: Let  $\mathcal{E}_i$  be linearly expressed by  $\{\mathcal{E}_{j_1}, \dots, \mathcal{E}_{j_k}\}$ . This implies that

$$\mathcal{E}_i = a_1\mathcal{E}_{j_1} + \dots + a_k\mathcal{E}_{j_k}$$

for some  $a_1, \dots, a_k$ . Since  $\{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$  is an admissible error vector set of  $\mathcal{Y}$ , it implies that for each  $i$ ,  $Y_i = \mathcal{C}_i + \mathcal{E}_i$ , where  $\mathcal{C}_i$  is some codeword. Then in mod  $\mathcal{C}$ , we have:

$$\begin{aligned} & Y_i - (a_1Y_{j_1} + \dots + a_kY_{j_k}) \\ &= (\mathcal{C}_i + \mathcal{E}_i) - a_1(\mathcal{C}_{j_1} + \mathcal{E}_{j_1}) - \dots - a_k(\mathcal{C}_{j_k} + \mathcal{E}_{j_k}) \\ &= (\mathcal{C}_i - a_1\mathcal{C}_{j_1} - \dots - a_k\mathcal{C}_{j_k}) + (\mathcal{E}_i - a_1\mathcal{E}_{j_1} - \dots - a_k\mathcal{E}_{j_k}) \\ &= (\mathcal{C}_i - a_1\mathcal{C}_{j_1} - \dots - a_k\mathcal{C}_{j_k}) + 0 \\ &= (\mathcal{C}_i - a_1\mathcal{C}_{j_1} - \dots - a_k\mathcal{C}_{j_k}) \end{aligned}$$

Now notice that  $\mathcal{C}_i, \mathcal{C}_{j_1}, \dots, \mathcal{C}_{j_k}$  are valid RS codewords belonging to  $\mathcal{C}$  and hence are encoded using polynomials of degree  $t_b + t_p$ . So from the linearity property of polynomials,  $\mathcal{C}_i - a_1\mathcal{C}_{j_1} - \dots - a_k\mathcal{C}_{j_k}$  will also be a valid RS codeword belonging to  $\mathcal{C}$  and hence is encoded using polynomial of degree  $t_b + t_p$ . So  $\mathcal{C}_i - a_1\mathcal{C}_{j_1} - \dots - a_k\mathcal{C}_{j_k} \in \mathcal{C}$ . Therefore,  $Y_i - (a_1Y_{j_1} + \dots + a_kY_{j_k}) \in \mathcal{C}$ . Thus if  $\mathcal{E}_i$  is linearly expressed by  $\{\mathcal{E}_{j_1}, \dots, \mathcal{E}_{j_k}\}$  then  $Y_i$  is linearly pseudo-expressed by  $\{Y_{j_1}, \dots, Y_{j_k}\}$ .

Next suppose that  $Y_i$  is linearly pseudo-expressed by  $\{Y_{j_1}, \dots, Y_{j_k}\}$ . Then in mod  $\mathcal{C}$ , we have

$$Y_i = (a_1Y_{j_1} + \dots + a_kY_{j_k}) \text{ mod } \mathcal{C} \quad (12.7)$$

for some non-zero  $a_1, \dots, a_k$ . The above equation can be written as

$$\begin{aligned} (\mathcal{C}_i + \mathcal{E}_i) &= \{a_1(\mathcal{C}_{j_1} + \mathcal{E}_{j_1}) + \dots + a_k(\mathcal{C}_{j_k} + \mathcal{E}_{j_k})\} \text{ mod } \mathcal{C} \\ \Rightarrow (\mathcal{C}_i - a_1\mathcal{C}_{j_1} - \dots - a_k\mathcal{C}_{j_k}) &+ (\mathcal{E}_i - a_1\mathcal{E}_{j_1} - \dots - a_k\mathcal{E}_{j_k}) \in \mathcal{C} \end{aligned}$$

Now  $\mathcal{C}_i, \mathcal{C}_{j_1}, \dots, \mathcal{C}_{j_k}$  are valid RS codewords, encoded using polynomials of degree  $t_b + t_p$ . So from the linearity property of polynomials, we have

$$(\mathcal{C}_i - a_1\mathcal{C}_{j_1} - \dots - a_k\mathcal{C}_{j_k}) \in \mathcal{C} \quad (12.8)$$

The linearity property of polynomials also implies:

$$(\mathcal{E}_i - a_1\mathcal{E}_{j_1} - \dots - a_k\mathcal{E}_{j_k}) \in \mathcal{C}$$

as the sum of a valid RS codeword (encoded using polynomial of degree  $t_b + t_p$ ) with another vector can be a valid RS codeword (encoded using polynomial of degree  $t_b + t_p$ ) only if the other vector is also a valid RS codeword (encoded using polynomial of degree  $t_b + t_p$ ). As the number of non-zero components in  $\mathcal{E}_i, \mathcal{E}_{j_1}, \dots, \mathcal{E}_{j_k}$  is at most  $t_b + t_f$ , the vector  $(\mathcal{E}_i - a_1\mathcal{E}_{j_1} - \dots - a_k\mathcal{E}_{j_k})$  will have at least  $n - (t_b + t_f) \geq t_b + t_p + 1$  zero components. However, in  $\mathcal{C}$ , there is only one codeword, namely 0-codeword (i.e. a tuple of length  $n$  containing all 0's)  $(0, \dots, 0)$ , which has at least  $t_b + t_p + 1$  zero components. This is because each element of  $\mathcal{C}$  represents  $n$  distinct points on a  $t_b + t_p$  degree polynomial and  $t_b + t_p + 1$  zero's uniquely define the zero polynomial  $f(x) = \sum_{i=0}^{t_b+t_p} 0x^i$ . These two facts together imply that the vector  $(\mathcal{E}_i - a_1\mathcal{E}_{j_1} - \dots - a_k\mathcal{E}_{j_k})$  is an all zero vector (0-codeword) which further implies that

$$\mathcal{E}_i = a_1\mathcal{E}_{j_1} + \dots + a_k\mathcal{E}_{j_k}$$

This means that if  $Y_i$  is linearly pseudo-expressed by  $\{Y_{j1}, \dots, Y_{jk}\}$ , then  $\mathcal{E}_i$  is linearly expressed by  $\{\mathcal{E}_{j1}, \dots, \mathcal{E}_{jk}\}$ .  $\square$

We next define *pseudo-span*.

**Definition 12.6 (Pseudo-Span [42])** We say that  $\{Y_{j1}, \dots, Y_{jk}\} \subset \mathcal{Y}$  pseudo-spans  $\mathcal{Y}$  if each  $Y_i \in \mathcal{Y}$  can be linearly pseudo-expressed by  $\{Y_{j1}, \dots, Y_{jk}\}$ . That is, each  $Y_i = (a_1 Y_{j1} + \dots + a_k Y_{jk}) \bmod \mathcal{C}$ , for some  $a_1, \dots, a_k$ .

**Definition 12.7 (Pseudo-Basis [42])** We say that  $\{Y_{j1}, \dots, Y_{jk}\} \subset \mathcal{Y}$  is a pseudo-basis of  $\mathcal{Y}$  if it is the minimum subset of  $\mathcal{Y}$  which pseudo-spans  $\mathcal{Y}$ .

**Definition 12.8 (Pseudo-Dimension [42])** If  $\{Y_{j1}, \dots, Y_{jk}\} \subset \mathcal{Y}$  is a pseudo-basis of  $\mathcal{Y}$  and  $k = |\{Y_{j1}, \dots, Y_{jk}\}|$ , then we say that  $\mathcal{Y}$  has pseudo-dimension  $k$ .

We now prove the following theorem:

**Theorem 12.9** Let  $\{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$  be an admissible error vector set of  $\mathcal{Y}$ . Then  $\mathcal{B}_e = \{\mathcal{E}_{j1}, \dots, \mathcal{E}_{jk}\} \subset \{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$  is a basis of the admissible error vector space  $\mathcal{E}$  iff  $\mathcal{B}_y = \{Y_{j1}, \dots, Y_{jk}\} \subset \mathcal{Y}$  is a pseudo-basis of  $\mathcal{Y}$  (Note that  $\mathcal{B}_e$  and  $\mathcal{B}_y$  have the same indices). In particular, the pseudo-dimension of  $\mathcal{Y}$  is equal to the dimension of  $\mathcal{E}$ .

PROOF: Suppose that  $\mathcal{B}_e$  is a basis of  $\mathcal{E}$ . This implies that  $\mathcal{B}_e$  is the minimum set which spans  $\mathcal{E}$ . Since  $\mathcal{B}_e$  spans  $\mathcal{E}$ , from Lemma 12.5,  $\mathcal{B}_y$  pseudo-spans  $\mathcal{Y}$ . Next we show that  $\mathcal{B}_y$  is the minimum subset of  $\mathcal{Y}$  which pseudo-spans  $\mathcal{Y}$ . On the contrary, assume that  $\mathcal{B}_y$  is not minimum. That is, suppose that there exists a smaller subset of  $\mathcal{Y}$  which pseudo-spans  $\mathcal{Y}$ . Then from Lemma 12.5, the corresponding subset of  $\{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$  also spans  $\mathcal{E}$ . However, this contradicts the fact that  $\mathcal{B}_e$  is a basis of  $\mathcal{E}$ . This implies that  $\mathcal{B}_y$  is the minimum subset of  $\mathcal{Y}$  which pseudo-spans  $\mathcal{Y}$ , which further implies that  $\mathcal{B}_y$  is the pseudo-basis of  $\mathcal{Y}$ .

Similarly, if  $\mathcal{B}_y$  is a pseudo-basis of  $\mathcal{Y}$  then  $\mathcal{B}_e$  is a basis of  $\mathcal{E}$ . Hence the pseudo-dimension of  $\mathcal{Y}$  is equal to the dimension of  $\mathcal{E}$ .  $\square$

Since the real error vector set is also an admissible error vector set, we obtain the following corollary of Theorem 12.9.

**Corollary 12.9.1** Let  $\{E_1, \dots, E_\gamma\}$  be the real error vector set of  $\mathcal{Y}$ . If  $\mathcal{B}_y = \{Y_{j1}, \dots, Y_{jk}\} \subset \mathcal{Y}$  is a pseudo-basis of  $\mathcal{Y}$ , then  $\mathcal{B}_e = \{E_{j1}, \dots, E_{jk}\} \subset \{E_1, \dots, E_\gamma\}$  is a basis of the real error vector space.

Let  $\{E_1, \dots, E_\gamma\}$  be the real error vector set of  $\mathcal{Y}$  and let  $\{C_1, \dots, C_\gamma\}$  be the corresponding original codewords which  $\mathbf{A}$  sent. Then define

$$CORRUPTED = \cup_{i=1}^{\gamma} \text{support}(E_i) \quad (12.9)$$

Then *CORRUPTED* is the set of wires that the adversary has corrupted in Byzantine and fail-stop fashion. We now state the following important theorem:

**Theorem 12.10** Let  $\mathcal{B}_y = \{Y_{j1}, \dots, Y_{jk}\}$  be the pseudo-basis of  $\mathcal{Y}$  and let  $C_{j1}, \dots, C_{jk}$  be the corresponding original codewords. Then

$$CORRUPTED = \cup_{i=1}^{i=k} \text{support}(Y_{ji} - C_{ji})$$

PROOF: From the definition of *CORRUPTED*, we get

$$CORRUPTED = \cup_{i=1}^{\gamma} support(E_i)$$

From Corollary 12.9.1, since  $\{Y_{j_1}, \dots, Y_{j_k}\}$  is the pseudo-basis of  $\mathcal{Y}$ , it implies that  $\{E_{j_1}, \dots, E_{j_k}\}$  is the basis of real error vector space. This implies that

$$\cup_{i=1}^{\gamma} support(E_i) = \cup_{i=1}^{i=k} support(E_{j_i})$$

The above relationship further implies that

$$\begin{aligned} CORRUPTED &= \cup_{i=1}^{\gamma} support(E_i) \\ &= \cup_{i=1}^{i=k} support(E_{j_i}) \\ &= \cup_{i=1}^{i=k} support(Y_{j_i} - X_{j_i}) \end{aligned}$$

□

**Theorem 12.11** *The pseudo-dimension of  $\mathcal{Y}$  is at most  $t_b + t_f$ .*

PROOF: The dimension of the real error vector space is at most  $t_b + t_f$  because the adversary can Byzantine and fail-stop corrupt at most  $t_b$  and  $t_f$  wires respectively. Hence from Theorem 12.9, the pseudo-dimension of  $\mathcal{Y}$  is at most  $t_b + t_f$ . □

From the above discussion, we find that if  $\mathbf{B}$  can correctly find the pseudo-basis of  $\mathcal{Y}$  and reliably sends it back to  $\mathbf{A}$ , then  $\mathbf{A}$  can identify the wires which are Byzantine and fail-stop corrupted after doing the local computation. In the next section, we present a polynomial time algorithm, which allows  $\mathbf{B}$  to find the pseudo-basis of  $\mathcal{Y}$ .

### 12.5.2 How to Find Pseudo-Basis

In [42], the authors have presented a polynomial time algorithm to find pseudo-basis against  $\mathcal{A}_{t_b}^{static}$ . We now extend the algorithm against mixed adversary.

We first present a polynomial time algorithm which checks whether  $Y$  can be linearly pseudo-expressed by  $\{B_1, \dots, B_k\}$ . For a non-zero  $\beta = (a_1, \dots, a_k) \in \mathbb{F}^k$ , we define  $X(\beta)$  as

$$X(\beta) = Y - (a_1 B_1 + \dots + a_k B_k). \quad (12.10)$$

It is clear that  $Y$  can be linearly pseudo-expressed by  $\{B_1, \dots, B_k\}$  iff there exists some non-zero  $\beta$  such that  $X(\beta) \in \mathcal{C}$ . The algorithm for checking whether  $Y$  can be linearly pseudo-expressed by  $\{B_1, \dots, B_k\}$  is presented in Fig. 12.4. The algorithm will output YES iff  $X(\beta) \in \mathcal{C}$  for some non-zero  $\beta$ . In the algorithm, each  $x_j(\beta)$  will be a linear expression of  $(a_1, \dots, a_k)$ , as  $Y, \mathcal{B}$  are known and  $\beta$  is unknown. Similarly, each coefficient of  $f_{\beta}(x)$  will be a linear expression of  $(a_1, \dots, a_k)$ . Hence each  $f_{\beta}(\alpha_j) = x_j(\beta)$  will give a linear equation on  $(a_1, \dots, a_k)$ . So we will get  $n - (t_b + t_p + 1) \geq t_b + t_f$  linear equations in  $(a_1, \dots, a_k)$ . Moreover, in our context,  $k$  will be at most  $t_b + t_f$ , as will be shown in the sequel. Thus if  $Y$  can be indeed linearly pseudo expressed by  $\mathcal{B}$ , then after solving these linear equations, some non-zero solution for  $\beta = (a_1, \dots, a_k)$  will be obtained and the algorithm will output YES.

We now finally present the polynomial time algorithm which takes the set of vectors  $\mathcal{Y}$  received by  $\mathbf{B}$  as input and finds in polynomial time, the pseudo-basis  $\mathcal{B} = \{Y_{j_1}, \dots, Y_{j_k}\} \subset \mathcal{Y}$ , pseudo-dimension  $k = |\mathcal{B}| \leq t_b + t_f$  and an index set

Figure 12.4: Algorithm to Check Whether  $Y$  Can be Linearly Pseudo-Expressed by  $\mathcal{B}$

**Algorithm Pseudo-Linear-Express**( $Y, \mathcal{B} = \{B_1, \dots, B_k\}$ )

1. Let  $\beta = (a_1, \dots, a_k)$ , where  $a_1, \dots, a_k$  are unknown variables.
2. Let  $X(\beta) = (x_1(\beta), \dots, x_n(\beta))$ , where  $X(\beta)$  is given by Equation 12.10.
3. By using Lagrange interpolation, construct a polynomial  $f_\beta(x)$  of degree  $t_b + t_p$  such that
 
$$f_\beta(\alpha_i) = x_i(\beta)$$
 for  $i = 1, \dots, t_b + t_p + 1$ .
4. Output YES iff the following set of linear equations on  $\beta$  has a non-zero solution:
 
$$f_\beta(\alpha_{t_b+t_p+2}) = x_{t_b+t_p+2}(\beta),$$

$$\vdots$$

$$f_\beta(\alpha_n) = x_n(\beta).$$
 Otherwise output NO.

$\mathcal{I} = \{j_1, \dots, j_k\} \subset \{1, \dots, \gamma\}$  containing the indices of the vectors selected in  $\mathcal{B}$ . The algorithm uses algorithm Pseudo-Linear-Express as a black-box and is provided in Fig. 12.5.

Figure 12.5: Algorithm to Find the Pseudo-Basis of  $\mathcal{Y}$

**Algorithm Find-Pseudo-Basis**( $\mathcal{Y}$ )

1. Let  $i = 1$  and  $\mathcal{B} = \emptyset$ .
2. While  $i \leq \gamma$  and  $|\mathcal{B}| < t_b + t_f$ , do:
  - (a) By using Algorithm Pseudo-Linear-Express, check whether  $Y_i$  can be linearly pseudo-expressed by  $\mathcal{B}$ . If NO, then add  $Y_i$  to  $\mathcal{B}$ .
  - (b) Set  $i \leftarrow i + 1$ .
3. Output  $\mathcal{B}$  as a pseudo-basis,  $k = |\mathcal{B}|$  as the pseudo-dimension and index set  $\mathcal{I}$ , containing the indices of the  $k$  vectors selected in  $\mathcal{B}$ .

**Theorem 12.12** *Algorithm Find-Pseudo-Basis correctly finds the pseudo-basis and pseudo-dimension of  $\mathcal{Y}$ .*

**Note 12.13 (Convention For Using Algorithm Find-Pseudo-Basis)** *In the rest of the chapter, we will use the notation  $(k, \mathcal{B}, \mathcal{I}) = \text{Find-Pseudo-basis}(\mathcal{Y})$  while invoking algorithm Find-Pseudo-Basis.*

### 12.5.3 Protocol 3-Optimal-PSMT-Mobile-Mixed Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + t_f + t_p + 1$  wires. We now present a three phase communication optimal PSMT protocol called 3-Optimal-PSMT-Mobile-Mixed, which securely sends a message containing  $n^2$  field elements by communicating  $\mathcal{O}(n^3)$  field elements tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Since  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is a special type of  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , the protocol will also work against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .

In the protocol,  $\mathbf{S}$  establishes a random, information theoretically secure one time pad of size  $n^2$  with  $\mathbf{R}$ . Once the pad is established,  $\mathbf{S}$  uses the pad to blind the message and sends the blinded message reliably to  $\mathbf{R}$ . Let  $\mathcal{C}$  denote the set of all possible RS codewords of length  $n = 2t_b + t_f + t_p + 1$  encoded using polynomials of degree  $(t_b + t_p)$  over  $\mathbb{F}$ . We now present protocol 3-Optimal-PSMT-Mobile-Mixed, which is given in Fig. 12.6. Recall that we are following Assumption 12.3 in protocol 3-Optimal-PSMT-Mobile-Mixed.

We now prove the properties of protocol 3-Optimal-PSMT-Mobile-Mixed.

**Lemma 12.14 (Perfect Reliability)** *In Protocol 3-Optimal-PSMT-Mobile-Mixed,  $\mathbf{R}$  will correctly recover  $m^{\mathbf{S}}$ .*

PROOF: First notice that  $\mathbf{R}$  will correctly receive the blinded message  $\Gamma$  and list *CORRUPTED*, as they are broadcasted by  $\mathbf{S}$ . Now to prove that  $\mathbf{R}$  correctly recovers the message  $m^{\mathbf{S}}$  sent by  $\mathbf{S}$ , we show that  $\mathbf{S}$  and  $\mathbf{R}$  shares the same pad  $Z$ .  $\mathbf{S}$  and  $\mathbf{R}$  will share  $Z$  if:

1.  $\mathcal{I}$  is same at both ends and
2.  $\mathbf{R}$  is able to correctly recover polynomials  $F_i^{\mathbf{S}}(x)$  for  $i \in \{1, \dots, P\} \setminus \mathcal{I}$ .

Since  $\mathbf{R}$  sends the triplet  $(\mathcal{B}, k, \mathcal{I})$  to  $\mathbf{S}$  by broadcasting,  $\mathcal{I}$  will be same at both ends. Now we show that irrespective of the behavior of  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ ,  $\mathbf{R}$  will always recover all the polynomials  $F_i^{\mathbf{S}}(x)$  for  $i \in \{1, \dots, P\} \setminus \mathcal{I}$ . Since  $\mathbf{S}$  correctly receives  $(\mathcal{B}, k, \mathcal{I})$ , *CORRUPTED* will contain all the wires which were corrupted in Byzantine and fail-stop fashion during **Phase I**.  $\mathbf{R}$  also correctly receives *CORRUPTED* from  $\mathbf{S}$ . Now ignoring the values received over the wires in *CORRUPTED* during **Phase I**,  $\mathbf{R}$  recovers all the polynomials with the remaining values. This is possible because each polynomial is of degree  $t_b + t_p$  and at least  $n - |\text{CORRUPTED}| \geq n - (t_b + t_f) = t_b + t_p + 1$  correct values on each polynomial, obtained over the wires in  $\{w_1, \dots, w_n\} \setminus \text{CORRUPTED}$  during **Phase I**, are available to  $\mathbf{R}$ , at the end of **Phase III**.  $\square$

**Lemma 12.15 (Perfect Security)** *In Protocol 3-Optimal-PSMT-Mobile-Mixed,  $m^{\mathbf{S}}$  will be information theoretically secure.*

PROOF: The message  $m^{\mathbf{S}}$  will be information theoretically secure from  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  if the pad  $Z$  is information theoretically secure. According to the protocol,  $Z$  contains  $F_i^{\mathbf{S}}(0)$  iff  $i \in \{1, \dots, P\} \setminus \mathcal{I}$ . Since  $(\mathcal{B}, k, \mathcal{I})$  was sent by  $\mathbf{R}$  by broadcasting, it may be eavesdropped by  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  during its transmission. But for remaining polynomials  $F_i^{\mathbf{S}}(x)$ 's where  $i \in \{1, \dots, P\} \setminus \mathcal{I}$ ,  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  will know at most  $t_b + t_p$  points by eavesdropping  $t_b + t_p$  wires during **Phase I**. Since the degree of each of these polynomials is  $t_b + t_p$ ,  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  will lack one point on these polynomials to uniquely interpolate them and hence each  $F_i^{\mathbf{S}}(0)$  with  $i \in \{1, \dots, P\} \setminus \mathcal{I}$  will be information theoretically secure.  $\square$

**Lemma 12.16 (Communication Complexity)** *Protocol 3-Optimal-PSMT-Mobile-Mixed sends a message  $m^{\mathbf{S}}$  containing  $\ell = n^2$  field elements by communicating  $\mathcal{O}(n^3) = \mathcal{O}\left(\frac{n\ell}{n-(2t_b+t_f+t_p)}\right) = \mathcal{O}(n\ell)$  field elements, tolerating  $\mathcal{A}_{(t_b,t_f,t_p)}^{mobile}$ . Moreover, the protocol is communication protocol.*

PROOF: During **Phase I**, **S** communicates  $P = n^2 + t_b + t_f$  codewords to **R** which requires communication of  $Pn = (n^2 + t_b + t_f).n = \mathcal{O}(n^3)$  field elements. During **Phase II**, **R** sends the triplet  $(\mathcal{B}, q, \mathcal{I})$  through all the wires. This incurs a communication cost of  $\mathcal{O}((t_b + t_f).n.n) = \mathcal{O}(n^3)$ . The communication complexity of **Phase III** for sending *CORRUPTED* and  $\Gamma$  is  $\mathcal{O}(n^2.n) = \mathcal{O}(n^3)$ . Hence overall communication complexity of the protocol is  $\mathcal{O}(n^3)$ .

From Theorem 11.5, any three phase PSMT tolerating  $\mathcal{A}_{(t_b,t_f,t_p)}^{static}$  must communicate  $\Omega(n^3)$  field elements to securely send a message containing  $\ell = n^2$  field elements, if **S** and **R** are connected by  $n = 2t_b + t_f + t_p + 1$  wires. Since  $\mathcal{A}_{(t_b,t_f,t_p)}^{mobile}$  is more powerful than  $\mathcal{A}_{(t_b,t_f,t_p)}^{static}$ , the same lower bound must hold against  $\mathcal{A}_{(t_b,t_f,t_p)}^{mobile}$ . As the total communication complexity of protocol 3-Optimal-PSMT-Mobile-Mixed is  $\mathcal{O}(n^3)$ , the protocol is a communication optimal PSMT protocol tolerating  $\mathcal{A}_{(t_b,t_f,t_p)}^{mobile}$ .  $\square$

## 12.6 Concluding Remarks and Open Problems

In this chapter, we completely resolved the issue of POSSIBILITY, FEASIBILITY and OPTIMALITY of PSMT in undirected synchronous network tolerating  $\mathcal{A}_{(t_b,t_f,t_p)}^{mobile}$ . These results are summarized in Fig. 12.7.

From Fig. 12.7, we find that protocol 9-Optimal-PSMT-Mobile-Mixed, as well as protocol 3-Optimal-PSMT-Mobile-Mixed is communication optimal only if the message contains  $\ell = \Theta(n)$  and  $\Theta(n^2)$  field elements respectively. Also, protocol 3-Optimal-PSMT-Mobile-Mixed requires three phases.

This leads to the following open problem:

**Open Problem 13** *Let **S** and **R** be connected by  $n = 2t_b + t_f + t_p + 1$  wires. Then does there exist a two phase PSMT protocol which securely sends a message containing  $\ell$  field elements, by communicating  $\mathcal{O}(n\ell)$  field elements, tolerating  $\mathcal{A}_{(t_b,t_f,t_p)}^{mobile}$ , for any value of  $\ell$ ?*

Figure 12.6: A Three Phase Communication Optimal PSMT Protocol Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ :  $n = 2t_b + t_f + t_p + 1, |m^S| = n^2$

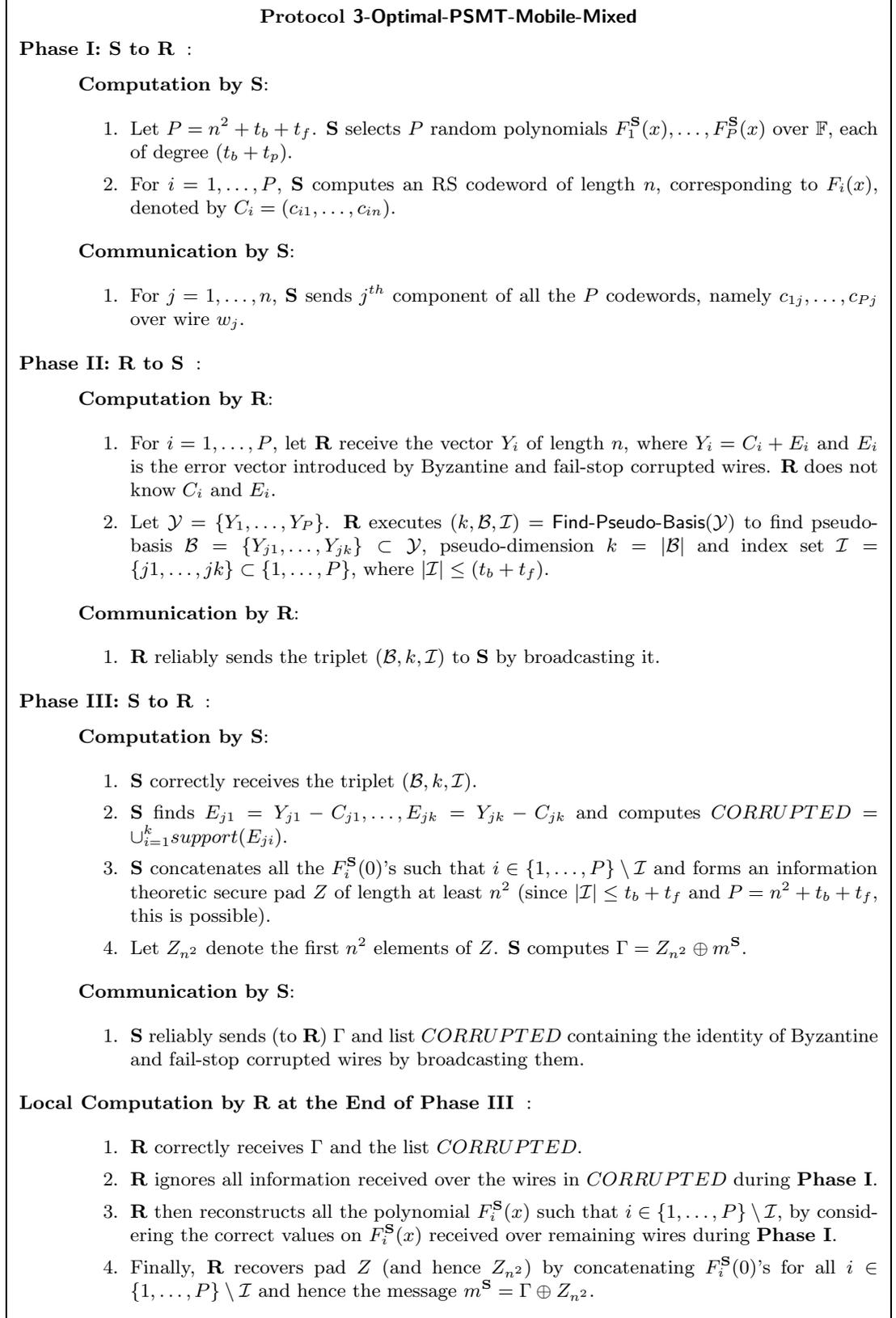


Figure 12.7: Summary of the Results for PSMT in Undirected Synchronous Network Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$

Number of Phases ( $r$ )	Connectivity Requirement ( $n$ )	Lower Bound on Communication Complexity	Upper Bound
$r = 1$	$n \geq 3t_b + t_f + t_p + 1$ Theorem 11.1	$\Omega\left(\frac{n\ell}{n-(3t_b+t_f+t_p)}\right)$ Theorem 11.3	Theorem 11.3
$r \geq 2$	$n \geq 2t_b + t_f + t_p + 1$ Theorem 11.4	$\Omega\left(\frac{n\ell}{n-(2t_b+t_f+t_p)}\right)$ Theorem 11.5	<p>Protocol            9-Optimal-PSMT-Mobile-Mixed:  <math>n = 2t_b + t_f + t_p + 1, \ell = \Theta(n)</math>            Communication            complexity = <math>\mathcal{O}(n\ell) = \mathcal{O}(n^2)</math></p> <p>Protocol            3-Optimal-PSMT-Mobile-Mixed:  <math>n = 2t_b + t_f + t_p + 1, \ell = \Theta(n^2)</math>            Communication            complexity = <math>\mathcal{O}(n\ell) = \mathcal{O}(n^3)</math></p>

## Chapter 13

# PSMT in Directed Networks Tolerating Static Byzantine Adversary

In this chapter, we study PSMT in *directed synchronous network tolerating threshold static Byzantine adversary*. As discussed in Chapter 7, in several practical scenarios, it is appropriate to model the underlying network as directed graphs. Motivated by this, Desmedt et al. [24] introduced the PSMT problem in directed network. In [24], the authors have given the necessary and sufficient condition for the existence of PSMT protocols in directed networks tolerating threshold static Byzantine adversary  $\mathcal{A}_{t_b}^{static}$ . The PSMT protocols of [24] were significantly improved in [62]. However recently in [88], the authors have shown that the PSMT protocols of [24] and [62] do not provide perfect secrecy. Specifically, they specified an adversary strategy against the PSMT protocols of [24, 62], which they called as *guessing attack*, which allows the adversary to gain some extra information about the secret message of  $\mathbf{S}$ . This re-opens the issue of POSSIBILITY of PSMT in directed networks.

In this chapter, we re-visit the PSMT problem in directed network tolerating  $\mathcal{A}_{t_b}^{static}$ . We design new PSMT protocols tolerating  $\mathcal{A}_{t_b}^{static}$ , which satisfies the characterization given by [24]. Our protocols are perfectly secure against guessing attack. Moreover, we also show that our protocols are communication optimal. For this, we derive non-trivial lower bound on the communication complexity of PSMT protocols in directed networks, tolerating  $\mathcal{A}_{t_b}^{static}$ . In short, we completely resolve the issue of POSSIBILITY, FEASIBILITY and OPTIMALITY of PSMT in directed synchronous networks tolerating  $\mathcal{A}_{t_b}^{static}$ . To the best of our knowledge, this is the first ever attempt in the literature of PSMT in directed networks.

We now briefly describe the network model and adversary settings used in this chapter.

### 13.1 Network Model and Adversary Settings

The network model and adversary settings used in this chapter are exactly same as in Chapter 7. Thus, we assume that there are  $n$  wires directed from  $\mathbf{S}$  to  $\mathbf{R}$ , denoted by  $f_1, \dots, f_n$  and  $u$  wires, directed from  $\mathbf{R}$  to  $\mathbf{S}$ , denoted by  $b_1, \dots, b_u$ . Moreover, the wires from  $\mathbf{S}$  to  $\mathbf{R}$  are node disjoint from the wires which are directed from  $\mathbf{R}$  to  $\mathbf{S}$ . The  $n$  wires from  $\mathbf{S}$  to  $\mathbf{R}$  are also called as *top band*, while the  $u$  wires from  $\mathbf{R}$  to  $\mathbf{S}$  are called as *bottom band*. We assume the presence of a static, threshold adversary  $\mathcal{A}_{t_b}^{static}$ ,

having *unbounded computing power*, who can control at most  $t_b$  wires, out of  $n + u$  wires in Byzantine fashion.

In the next section, we recall the existing results for PSMT in directed synchronous networks tolerating  $\mathcal{A}_{t_b}^{static}$ .

## 13.2 Existing Results for PSMT in Directed Network

We now summarize the existing results for PSMT in directed networks tolerating  $\mathcal{A}_{t_b}^{static}$ .

1. In [24], it is shown that any PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff there exists  $n = \max(3t_b - 2u + 1, 2t_b + 1)$  wires in the top band. To show the sufficiency of this condition, the authors in [24] presented an *inefficient* PSMT protocol. Moreover, the authors presented an efficient PSMT protocol provided there exists  $n = \max(3t_b - u + 1, 2t_b + 1)$  wires in the top band.
2. The PSMT protocols of [24] were significantly improved by Patra et al. [62], who presented an efficient three phase PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , in the presence of  $n = \max(3t_b - 2u + 1, 2t_b + 1)$  wires in the top band. The authors in [62] presented another three phase efficient PSMT protocol with  $n = \max(3t_b - u + 1, 2t_b + 1)$  wires in the top band, which significantly improves the efficient PSMT protocol of [24] in the same settings. Independently, in [87], the authors presented an efficient PSMT protocol in the presence of  $n = \max(3t_b - 2u + 1, 2t_b + 1)$  wires in the top band.
3. In [62], the authors claimed that for the existence of any two phase PSMT protocol in directed network tolerating  $\mathcal{A}_{t_b}^{static}$ , there should exist  $2t_b + 1$  wires in the top band, as well as in the bottom band (the wires in the top band and the bottom band need not be disjoint in this case).
4. In [88], the authors showed that none of the PSMT protocol presented in [24, 62], tolerating  $\mathcal{A}_{t_b}^{static}$ , provides perfect security. Specifically, the authors in [88] presented guessing attack against the PSMT protocols of [24, 62], tolerating  $\mathcal{A}_{t_b}^{static}$  and showed that the guessing attack allows the adversary to get extra information about the secret message.

In the next section, we give an overview of our results presented in this chapter.

## 13.3 Overview of Our Results for PSMT in Directed Networks

In this chapter, we show the following:

1. We show that the condition for two phase PSMT in directed network as given in [62] is sufficient but *not necessary*. On the other hand, the condition for PSMT in directed network as given in [24] is necessary but *not sufficient* for two phase PSMT. This brings forth the question that what is the necessary and sufficient condition for two phase PSMT in directed network tolerating  $\mathcal{A}_{t_b}^{static}$ . We settle this question by showing that two phase PSMT in directed network tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff there exists  $n = \max(3t_b - u + 1, 2t_b + 1)$  wires, instead of  $n = \max(3t_b - 2u + 1, 2t_b + 1)$  wires in the top band. We also design a two phase

PSMT protocol called **2-Optimal-PSMT-Static-Byzantine-Directed** in the presence of  $\max(3t_b - u + 1, 2t_b + 1)$  wires in the top band. Furthermore, we show that our two phase PSMT protocol is communication optimal. For this, we derive non-trivial lower bound on the communication complexity of two phase PSMT in directed network tolerating  $\mathcal{A}_{t_b}^{static}$ . Specifically, we show that any two phase PSMT protocol in directed network which securely sends a message  $m^{\mathbf{S}} \in \mathbb{F}^\ell$  containing  $\ell$  field elements against  $\mathcal{A}_{t_b}^{static}$  must communicate:

- (a)  $\Omega\left(\frac{N\ell}{N-3t_b}\right)$  field elements where  $0 < u \leq t_b$ ,  $n \geq 3t_b - u + 1$  and  $N = n + u \geq 3t_b + 1$ .
- (b)  $\Omega\left(\frac{n\ell}{n-2t_b}\right)$  field elements where  $u > t_b$  and  $n \geq 2t_b + 1$ .

2. We then propose two new PSMT protocols called **3-Optimal-PSMT-Static-Byzantine-Directed** and **6-Optimal-PSMT-Static-Byzantine-Directed**, which are secure against guessing attack of [88] and which requires  $n = \max(3t_b - 2u + 1, 2t_b + 1)$  wires in the top band, thus satisfying the characterization of PSMT in directed network given by [24]. Specifically, protocol **3-Optimal-PSMT-Static-Byzantine-Directed** works in the presence of  $0 < u \leq \frac{t_b}{2}$  and  $n = 3t_b - 2u + 1$  wires in the bottom and top band respectively. The protocol securely sends a message  $m^{\mathbf{S}}$  containing  $\ell = n^2u$  field elements by communicating  $\mathcal{O}(n^3u) = \mathcal{O}(n\ell)$  field elements.

On the other hand protocol **6-Optimal-PSMT-Static-Byzantine-Directed** works in the presence of  $n = 2t_b + 1$  and  $\frac{t_b}{2} < u \leq t_b$  wires in the top and bottom band respectively. The protocol securely sends a message  $m^{\mathbf{S}}$  containing  $\ell = n^2u$  field elements by communicating  $\mathcal{O}\left(\frac{n^3u}{2u-t_b}\right) = \mathcal{O}\left(\frac{n\ell}{2u-t_b+1}\right)$  field elements. Interestingly, when  $u = \frac{t_b}{2} + \Theta(t)$ , then protocol **6-Optimal-PSMT-Static-Byzantine-Directed** securely sends  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements.

3. We also show that protocol **3-Optimal-PSMT-Static-Byzantine-Directed** and **6-Optimal-PSMT-Static-Byzantine-Directed** are communication optimal. For this, we derive non-trivial lower bound on the communication complexity of three or more phase PSMT protocol in directed networks tolerating  $\mathcal{A}_{t_b}^{static}$ . Specifically, we show that if there exists  $u$  wires in the bottom band and  $n = \max(3t_b - 2u + 1, 2t_b + 1)$  wires in the top band, then any three or more phase PSMT protocol that securely sends a message  $m^{\mathbf{S}}$  containing  $\ell$  field elements from  $\mathbb{F}$  tolerating  $\mathcal{A}_{t_b}^{static}$  must communicate:

- (a)  $\Omega\left(\frac{n\ell}{n-(3t_b-2u)}\right)$  field elements when  $0 < u \leq t_b$ ;
- (b)  $\Omega(\ell)$  field elements when  $u > t_b$ .

**Remark 13.1 (On the Non-Zero Value of  $u$ )** *All the results and protocols that are discussed in this chapter considers  $u > 0$ . If  $u = 0$ , then all the wires are directed only from  $\mathbf{S}$  to  $\mathbf{R}$ . In this case, only single phase PSMT is possible. However, the issue of POSSIBILITY, FEASIBILITY and OPTIMALITY of single phase PSMT tolerating  $\mathcal{A}_t^{static}$  is already settled in [28, 81, 30] (see Chapter 9).*

## 13.4 Tools Used in Our Protocols

In this chapter, we will use protocol **3-Optimal-PRMT-Static-Byzantine-Directed** (see Fig. 7.2) as a black-box. Protocol **3-Optimal-PRMT-Static-Byzantine-Directed** is a communication optimal PRMT protocol which reliably sends a message containing  $\ell = \Theta(nt_b)$

field elements by communicating  $\mathcal{O}(nt_b)$  field elements tolerating  $\mathcal{A}_{t_b}^{static}$ , provided there exists  $n = 2t_b + 1$  wires in the top band and  $u$  wires in the bottom band, such that  $n - 2t_b + 2u = \Theta(n)$ .

In this chapter, we also use the concept of pseudo-basis, which was discussed in Section 12.5.1. In section 12.5.1, we explained the idea of pseudo-basis and pseudo-dimension in the context of mixed corruption. However, the same ideas will also for only Byzantine corruption. More specifically, let  $\mathbf{A}$  and  $\mathbf{B}$  be two specific nodes, which are connected by  $L$  wires denoted by  $w_1, \dots, w_L$ , of which at most  $t_b$  wires can be corrupted in Byzantine fashion by  $\mathcal{A}_{t_b}^{static}$ . Let  $\mathcal{C}$  be the set of all possible RS codewords of length  $L$  over  $\mathbb{F}$ , such that the hamming distance [45] of code  $\mathcal{C}$  is at least  $t_b + 1$ . This implies that each codeword in  $\mathcal{C}$  is RS encoded using polynomial of degree at most  $\deg = L - (t_b + 1)$ . Moreover, in our PSMT protocols,  $\deg$  will be at least  $t_b$ . Now suppose  $\mathbf{A}$  sends several codewords, say  $\gamma$  codewords  $C_1, \dots, C_\gamma \in \mathcal{C}$  over the  $L$  wires by transmitting the  $i^{th}$  component of all the codewords over  $i^{th}$  wire  $w_i$ . Let  $\mathbf{B}$  receive the vectors  $\mathcal{Y} = \{Y_1 \dots, Y_\gamma\}$  over the  $L$  wires, such that for  $i = 1, \dots, \gamma$ ,

$$Y_i = C_i + E_i, \quad (13.1)$$

where

$$E_i = (e_{i1}, \dots, e_{iL})$$

is an error vector introduced by the adversary. Then by using Algorithm Find-Pseudo-Basis (see Fig. 12.5),  $\mathbf{B}$  will be able to compute the pseudo-basis and pseudo-dimension of  $\mathcal{Y}$ . Moreover, pseudo-dimension of  $\mathcal{Y}$  will be at most  $t_b$ . Specifically, algorithm Find-Pseudo-Basis will output the pseudo-basis  $\mathcal{B} \subset \mathcal{Y}$ , the pseudo-dimension  $k = |\mathcal{B}| \leq t_b$  and index set  $\mathcal{I}$ , containing the indices of the vectors from  $\mathcal{Y}$  which are listed in  $\mathcal{B}$ . Moreover, if  $\mathcal{B}_y = \{Y_{j1}, \dots, Y_{jk}\} \subset \mathcal{Y}$  is the pseudo-basis of  $\mathcal{Y}$  and if  $C_{j1}, \dots, C_{jk}$  are the corresponding original codewords, then

$$CORRUPTED = \cup_{i=1}^{i=k} support(Y_{ji} - C_{ji}),$$

where  $CORRUPTED$  is the set of wires which are corrupted by  $\mathcal{A}_{t_b}^{static}$ .

In the next section, we discuss about two phase PSMT in directed network tolerating  $\mathcal{A}_{t_b}^{static}$ .

## 13.5 Two Phase PSMT in Directed Synchronous Network Tolerating $\mathcal{A}_{t_b}^{static}$

In this section, we prove the necessary and sufficient condition for the existence of any two phase PSMT in directed network tolerating  $\mathcal{A}_{t_b}^{static}$ . We then derive the lower bound on the communication complexity of any two phase PSMT protocol in directed networks tolerating  $\mathcal{A}_{t_b}^{static}$ . We then show that the bound is *asymptotically tight* by designing a two phase communication optimal PSMT protocol in directed network.

### 13.5.1 Characterization of Two Phase PSMT in Directed Network

Before proceeding further, we recall the following theorems from [24] and [62].

**Theorem 13.2 ([24, 87])** *PSMT in directed network tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff there exists  $n = \max(3t_b - 2u + 1, 2t_b + 1)$  wires in the top band.*

**Theorem 13.3 ([62])** *Two phase PSMT in directed network tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff there exists  $n \geq 2t_b + 1$  wires in the top band and  $u \geq 2t_b + 1$  wires in the bottom band.*

We now show that the condition given in Theorem 13.2 is necessary for two phase PSMT in directed network tolerating  $\mathcal{A}_{t_b}^{static}$ , but *not sufficient*. On the other hand, the condition given in Theorem 13.3 is sufficient for two phase PSMT in directed network tolerating  $\mathcal{A}_{t_b}^{static}$ , but *not necessary*. This brings forth the following question:

*What is the necessary and sufficient condition for the existence of any two phase PSMT in directed network tolerating  $\mathcal{A}_{t_b}^{static}$ ?*

We resolve the above question by proving the following theorem:

**Theorem 13.4** *Let there are disjoint set of  $n$  wires in the top band and  $u$  wires in the bottom band such that  $\mathcal{A}_{t_b}^{static}$  controls at most  $t_b$  of these  $n + u$  wires. Then there exists a two phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$  iff  $n \geq \max(3t_b - u + 1, 2t_b + 1)$ .*

**PROOF: Sufficiency:** Let  $n = \max(3t_b - u + 1, 2t_b + 1)$  and  $u > 0$ . Also let  $\delta = \max(u, t_b)$  and  $N = n + u$ . We now design a two phase PSMT protocol called 2-Optimal-PSMT-Static-Byzantine-Directed, which securely sends a message  $m^{\mathbf{S}}$ , containing  $\ell = (\delta + 1 - t_b)$  field elements by communicating  $(N + n(\delta + 1 - t_b))$  field elements. In 2-Optimal-PSMT-Static-Byzantine-Directed,  $\mathcal{C}$  is the set of all possible RS codewords of length  $N$ , encoded using all possible polynomials of degree  $\delta$  over  $\mathbb{F}$ , for fixed  $\alpha_1, \dots, \alpha_{n+u}$ . Here  $\alpha_i$  is associated with wire  $f_i$  for  $i = 1, \dots, n$  and  $\alpha_{n+i}$  is associated with  $b_i$  for  $i = 1, \dots, u$ . The hamming distance [45] between any two codeword in  $\mathcal{C}$  is  $N - \delta = n + u - \delta \geq 2t_b + 1$ .

Informally, 2-Optimal-PSMT-Static-Byzantine-Directed works as follows:  $\mathbf{S}$  and  $\mathbf{R}$  communicate with each other to agree on a random polynomial of degree  $\delta$ , ensuring that  $\mathcal{A}_{t_b}^{static}$  knows  $t_b$  points on it. Once this is done, both  $\mathbf{S}$  and  $\mathbf{R}$  generates a common information theoretic secure pad of length  $(\delta + 1 - t_b)$ , which is completely unknown to  $\mathcal{A}_{t_b}^{static}$ . Then,  $\mathbf{S}$  blinds the message with the pad and reliably sends the blinded message to  $\mathbf{R}$ . The protocol is formally presented in Fig. 13.1.

We now prove the properties of protocol 2-Optimal-PSMT-Static-Byzantine-Directed.

**Theorem 13.5 (Perfect Reliability)** *In protocol 2-Optimal-PSMT-Static-Byzantine-Directed,  $\mathbf{R}$  will correctly recover  $m^{\mathbf{S}}$  at the end of Phase II.*

**PROOF:** From the protocol, it is clear that  $C^{\mathbf{R}}$  and  $C^{\mathbf{S}}$  will differ in at most  $t_b$  locations. Now there are following two cases:

1.  $u < t_b$ : This implies that  $\delta = t_b$  and  $n = 3t_b - u + 1$  and hence  $N = 3t_b + 1$ . So substituting  $N = 3t_b + 1$  and  $k = \delta + 1 = t_b + 1$  in Theorem 2.19, we find that  $\mathbf{R}$  will correctly recover  $F^{\mathbf{S}}(x)$  and hence  $C^{\mathbf{S}}$  after executing RS-DEC( $N, C^{\mathbf{R}}, t_b, 0, \delta + 1$ ). Thus both  $\mathbf{S}$  and  $\mathbf{R}$  will agree on  $C^{\mathbf{S}}$  and hence on  $Z^{\mathbf{S}}$ . Since  $\Gamma^{\mathbf{S}}$  is broadcasted over at least  $2t_b + 1$  wires,  $\mathbf{R}$  will correctly receive  $\Gamma^{\mathbf{R}} = \Gamma^{\mathbf{S}}$  and hence recover  $m^{\mathbf{S}} = \Gamma^{\mathbf{R}} \oplus Z^{\mathbf{R}}$ .

2.  $u \geq t_b$ : This implies that  $\delta = u$  and  $n = 2t_b + 1$  and hence  $N = 2t_b + 1 + u$ . So substituting  $N = 2t_b + 1 + u$  and  $k = \delta + 1 = u + 1$  in Theorem 2.19, we find that  $\mathbf{R}$  will correctly recover  $F^{\mathbf{S}}(x)$  and hence  $C^{\mathbf{S}}$  after executing RS-DEC( $N, C^{\mathbf{R}}, t_b, 0, \delta + 1$ ). The proof now follows using similar argument as above.  $\square$

Figure 13.1: A Two Phase Communication Optimal PSMT Protocol Tolerating  $\mathcal{A}_{t_b}^{static}$

**Protocol 2-Optimal-PSMT-Static-Byzantine-Directed**

**Phase I: R to S:** **R** selects a random vector  $V^{\mathbf{R}} = (v_1^{\mathbf{R}}, \dots, v_u^{\mathbf{R}})$  over  $\mathbb{F}$  and sends  $v_i^{\mathbf{R}}$  to **S** along wire  $b_i$ , for  $i = 1, \dots, u$ .

**Phase II: S to R:**

1. Let **S** receive  $V^{\mathbf{S}}$ . **S** then selects a codeword  $C^{\mathbf{S}}$  from  $\mathcal{C}$  such that the last  $u$  components of  $C^{\mathbf{S}}$  are same as  $V^{\mathbf{S}}$ . This is always possible because  $\delta \geq u$  and every RS codeword in  $\mathcal{C}$  corresponds to a unique  $\delta$  degree polynomial. Let  $C^{\mathbf{S}}$  corresponds to polynomial  $F^{\mathbf{S}}(x)$  of degree  $\delta$ .
2. For  $i = 1, \dots, n$ , **S** sends the  $i^{th}$  component of  $C^{\mathbf{S}}$  over wire  $f_i$  in *top band*.
3. **S** computes  $\Gamma^{\mathbf{S}} = m^{\mathbf{S}} \oplus Z^{\mathbf{S}}$  where  $Z^{\mathbf{S}} = \text{EXTRAND}_{\delta+1, \delta+1-t_b}(C_{(\delta+1)}^{\mathbf{S}})$  and  $C_{(\delta+1)}^{\mathbf{S}}$  denotes the first  $\delta+1$  components of  $C^{\mathbf{S}}$ . /\* For the details of algorithm EXTRAND, see Fig. 9.2 \*/
4. **S** broadcasts the blinded message  $\Gamma^{\mathbf{S}}$  over the entire *top band* and terminates the protocol.

**Local Computation by R At The End of Phase II:**

1. After receiving information over the *top band*, **R** possesses a vector  $C^{\mathbf{R}}$  of length  $N$ , where  $N = n + u$ , by combining the values received over the top band and values sent over the bottom band.
2. **R** executes  $\text{RS-DEC}(N, C^{\mathbf{R}}, t_b, 0, \delta+1)$  and recovers  $F^{\mathbf{R}}(x) = F^{\mathbf{S}}(x)$  and hence  $C^{\mathbf{S}}$ .
3. **R** computes  $Z^{\mathbf{R}} = Z^{\mathbf{S}}$  in the same way as done by **S**.
4. **R** correctly receives  $\Gamma^{\mathbf{R}} = \Gamma^{\mathbf{S}}$  and recovers  $m^{\mathbf{R}} = m^{\mathbf{S}}$  by computing  $m^{\mathbf{R}} = \Gamma^{\mathbf{R}} \oplus Z^{\mathbf{R}}$ .

**Theorem 13.6 (Perfect Security)** *Protocol 2-Optimal-PSMT-Static-Byzantine-Directed is a PSMT protocol.*

PROOF: In the protocol,  $\mathcal{A}_{t_b}^{static}$  will know at most  $t_b$  points on  $F^{\mathbf{S}}(x)$ . However, the degree of  $F^{\mathbf{S}}(x)$  is  $\delta$ . Thus adversary will lack  $\delta + 1 - t_b$  points to uniquely interpolate  $F^{\mathbf{S}}(x)$ . This implies that  $\delta + 1 - t_b$  components of  $C_{\delta+1}^{\mathbf{S}}$  will be information theoretically secure. So from the properties of EXTRAND, the pad  $Z^{\mathbf{S}}$  will be information theoretically secure. This further implies that  $m^{\mathbf{S}}$  will be information theoretically secure, as  $m^{\mathbf{S}}$  is blinded with the pad  $Z^{\mathbf{S}}$ .  $\square$

Protocol 2-Optimal-PSMT-Static-Byzantine-Directed, along with Theorem 13.5 and Theorem 13.6 shows the sufficiency of the condition given in Theorem 13.4. We now proceed to the necessity proof of Theorem 13.4.

**Necessity:** The necessity proof is divided into two cases: (a)  $0 < u \leq t_b$  and (b)  $u > t_b$ . If  $u > t_b$ , then the necessary condition says that there should exist  $n = 2t_b + 1$  wires in the top band. From Theorem 7.1,  $n = 2t_b + 1$  wires from  $\mathbf{S}$  to  $\mathbf{R}$  are necessary for any PRMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ . So it is obviously necessary for PSMT.

Next we show that if  $0 < u \leq t_b$ , then  $n = 3t_b - u + 1$  wires in the *top band* are necessary for the existence of any two phase PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ . The proof is by contradiction. So assume that there exists a two phase PSMT protocol with  $0 < u \leq t_b$  wires in the *bottom band* and  $n = 3t_b - u$  wires in the *top band*, tolerating  $\mathcal{A}_{t_b}^{static}$ . Let  $\Pi^{2Phase}$  be an *execution* of the two phase PSMT protocol where  $\mathbf{S}$  sends message  $m$ . Let  $\mathcal{A}_{t_b}^{2Phase}$  be an adversarial strategy in  $\Pi^{2Phase}$ . Given  $\Pi^{2Phase}$  and  $\mathcal{A}_{t_b}^{2Phase}$ , we show that there exist an *execution*  $\Pi^{1Phase}$  of a single phase PSMT protocol over  $N = n + u = 3t_b$  wires from  $\mathbf{S}$  to  $\mathbf{R}$  and an adversarial strategy  $\mathcal{A}_{t_b}^{1Phase}$  in  $\Pi^{1Phase}$ , such that the views of  $\mathbf{S}$  and  $\mathbf{R}$  in  $\Pi^{2Phase}$  are identical to the views of  $\mathbf{S}$  and  $\mathbf{R}$  (respectively) in  $\Pi^{1Phase}$ . Now by the property of two phase PSMT,  $\mathbf{R}$  will recover  $m$  in  $\Pi^{2Phase}$  tolerating any strategy  $\mathcal{A}_{t_b}^{2Phase}$ . Since the views are identical,  $\mathbf{R}$  should also recover  $m$  in  $\Pi^{1Phase}$  tolerating  $\mathcal{A}_{t_b}^{1Phase}$ . But by Theorem 9.1, we show that  $\mathbf{R}$  can not recover  $m$  in  $\Pi^{1Phase}$  tolerating  $\mathcal{A}_{t_b}^{1Phase}$ . This in turn implies that  $\mathbf{R}$  will fail to recover  $m$  in  $\Pi^{2Phase}$  tolerating  $\mathcal{A}_{t_b}^{2Phase}$ , thus showing a contradiction.

We now describe the executions  $\Pi^{2Phase}$ ,  $\Pi^{1Phase}$  and the adversary strategies  $\mathcal{A}_{t_b}^{2Phase}$  and  $\mathcal{A}_{t_b}^{1Phase}$ . The random coin flips of  $\mathbf{S}$ ,  $\mathbf{R}$  and  $\mathcal{A}_{t_b}^{static}$  in  $\Pi^{2Phase}$  as well as in  $\Pi^{1Phase}$  are  $\mathcal{R}^{\mathbf{S}}$ ,  $\mathcal{R}^{\mathbf{R}}$  and  $\mathcal{R}^{\mathcal{A}}$  respectively. Since  $\Pi^{2Phase}$  is an instance of a two phase PSMT, without loss of generality, the computation and communication during  $\Pi^{2Phase}$  are as follows:

1. **Phase I: R to S:**  $\mathbf{R}$  uses  $\mathcal{R}^{\mathbf{R}}$  to generate  $\beta_1, \dots, \beta_u$  and sends  $\beta_i$  to  $\mathbf{S}$  through wire  $b_i, 1 \leq i \leq u$ .
2. **Phase II: S to R:** Let  $\mathbf{S}$  receive  $\beta'_i$  through wire  $b_i$ . Based on the received information, message  $m$  and coin flip  $\mathcal{R}^{\mathbf{S}}$ ,  $\mathbf{S}$  computes  $\alpha_1, \alpha_2, \dots, \alpha_n$  and sends  $\alpha_i$  to  $\mathbf{R}$  through wire  $f_i, 1 \leq i \leq n$ .
3. **Computation by R at the end of Phase II:** Let  $\mathbf{R}$  receive  $\alpha'_i$  through wire  $f_i$ . Thus the view of  $\mathbf{R}$  at the end of **Phase II** is  $[\alpha'_1, \dots, \alpha'_n, \beta_1, \dots, \beta_u]$ , while view of  $\mathbf{S}$  is  $[\alpha_1, \dots, \alpha_n, \beta'_1, \dots, \beta'_u]$ .  $\mathbf{R}$  performs local computation according to the protocol specification and correctly recovers  $m$ .

Now we present  $\Pi^{1Phase}$  where there exists  $N = n + u = 3t_b$  wires  $w_1, \dots, w_N$  from  $\mathbf{S}$  to  $\mathbf{R}$ .

1. **Phase I: S to R:**  $\mathbf{S}$  uses  $\mathcal{R}^{\mathbf{S}}$  to generate  $\beta'_1, \dots, \beta'_u$  (which he can do with non-zero probability). Now assuming that  $\beta'_1, \dots, \beta'_u$  would have been received through the bottom band in  $\Pi^{2Phase}$ ,  $\mathbf{S}$  performs the same computation as in  $\Pi^{2Phase}$  and generates  $\alpha_1, \dots, \alpha_n$ . Finally,  $\mathbf{S}$  sends  $\alpha_i$  to  $\mathbf{R}$  through wire  $w_i, 1 \leq i \leq n$  and  $\beta'_i$  through wire  $w_{n+i}, 1 \leq i \leq u$ .
2. **Computation by R at the end of Phase I:** Let  $\mathbf{R}$  receive  $\alpha''_i$  through wire  $w_i, 1 \leq i \leq n$  and  $\beta''_i$  through wire  $w_{n+i}, 1 \leq i \leq u$ . Now  $\mathbf{R}$  performs the same computation as in  $\Pi^{2Phase}$  to recover  $m$ .

Now consider the following strategy  $\mathcal{A}_{t_b}^{2Phase}$  in  $\Pi^{2Phase}$ :  $\mathcal{A}_{t_b}^{static}$  corrupts entire bottom band and first  $t_b - u$  wires from top band and ensures that  $\beta'_i \neq \beta_i$  for  $1 \leq i \leq u$  and  $\alpha'_i \neq \alpha_i$  for  $1 \leq i \leq t_b - u$ . So, the views of  $\mathbf{S}$  and  $\mathbf{R}$  are  $(\alpha_1, \dots, \alpha_{t_b - u}, \alpha_{t_b - u + 1}, \dots, \alpha_n, \beta'_1, \dots, \beta'_u)$  and  $(\alpha'_1, \dots, \alpha'_{t_b - u}, \alpha_{t_b - u + 1}, \dots, \alpha_n, \beta_1, \dots, \beta_u)$  respectively.

Now consider the following strategy  $\mathcal{A}_{t_b}^{1Phase}$  in  $\Pi^{1Phase}$ :  $\mathcal{A}_{t_b}^{static}$  corrupts last  $u$  wires and first  $t_b - u$  wires and ensures that  $\beta_i'' = \beta_i$  for  $1 \leq i \leq u$  and  $\alpha_i'' = \alpha_i'$ , for  $1 \leq i \leq (t_b - u)$ . Since all other wires are honest, it holds that  $\alpha_i'' = \alpha_i$  for  $t_b - u + 1 \leq i \leq n$ . Hence in this case, the views of  $\mathbf{S}$  and  $\mathbf{R}$  will be exactly same as in the execution  $\Pi^{2Phase}$  where  $\mathcal{A}_{t_b}^{2Phase}$  is the adversary strategy. If in  $\Pi^{2Phase}$ ,  $\mathbf{R}$  is able to recover  $m$ , same should hold for  $\Pi^{1Phase}$ . But from Theorem 9.1, single phase PSMT over  $3t_b$  wires is impossible tolerating  $t_b$  faults done by  $\mathcal{A}_{t_b}^{static}$ . Hence by the argument given before, this leads to a contradiction to our assumption that  $\Pi^{2Phase}$  is an execution of two phase PSMT. Therefore for  $0 < u \leq t_b$ , the condition  $n \geq 3t_b - u + 1$  should hold for two phase PSMT. This completes the necessity proof.  $\square$

### 13.5.2 Lower Bound on Communication Complexity of Two Phase PSMT

We now derive the lower bound on the communication complexity of two phase PSMT in directed networks, tolerating  $\mathcal{A}_{t_b}^{static}$ . We then show that this bound is asymptotically tight. The lower bound on the communication complexity is given by the following theorem:

**Theorem 13.7** *Suppose there exists  $u$  wires in the bottom band and  $n = \max(3t_b - u + 1, 2t_b + 1)$  wires in the top band. Then any two phase PSMT protocol which securely sends a message  $m \in \mathbb{F}^\ell$  containing  $\ell$  field elements against  $\mathcal{A}_{t_b}^{static}$  must communicate:*

1.  $\Omega\left(\frac{N\ell}{N-3t_b}\right)$  field elements where  $0 < u \leq t_b$ ,  $n \geq 3t_b - u + 1$  and  $N = n + u \geq 3t_b + 1$ .
2.  $\Omega\left(\frac{n\ell}{n-2t_b}\right)$  field elements where  $u > t_b$  and  $n \geq 2t_b + 1$ .

PROOF: We first prove part 1 of this theorem. This proof is heavily based on the necessity proof of Theorem 13.4. Following the same line of argument, we can show that when  $n = 3t_b - u + 1$  and  $0 < u \leq t_b$ , then for every possible pair of  $\Pi^{2Phase}$  and  $\mathcal{A}_{t_b}^{2Phase}$  there exist a pair  $\Pi^{1Phase}$  and  $\mathcal{A}_{t_b}^{1Phase}$  (with non-zero probability) such that the view of  $\mathbf{S}$  and  $\mathbf{R}$  are same in both the scenarios. It is easy to see that the communication cost are also same in  $\Pi^{1Phase}$  and  $\Pi^{2Phase}$ . It implies that for every two phase PSMT protocol sending  $m^{\mathbf{S}}$  with  $n \geq 3t_b - u + 1$  and  $0 < u \leq t_b$  wires in *top* and *bottom* band respectively, there exist a single phase PSMT sending  $m^{\mathbf{S}}$  with  $N = n + u$  wires (from  $\mathbf{S}$  to  $\mathbf{R}$ ) with same communication cost. Now any single phase PSMT protocol sending  $m^{\mathbf{S}}$  over  $N \geq 3t_b + 1$  wires must communicate  $\Omega\left(\frac{N\ell}{N-3t_b}\right)$  field elements (see Theorem 9.2). Hence any two phase PSMT protocol must communicate  $\Omega\left(\frac{N\ell}{N-3t_b}\right)$  field elements for sending  $m^{\mathbf{S}}$ .

We now proceed to prove part 2 of the theorem. Any PSMT protocol has to deliver the message correctly. Thus any PSMT protocol is also a PRMT protocol. Now neglecting the communication from  $\mathbf{R}$  to  $\mathbf{S}$ , any two phase PRMT can be reduced to single phase PRMT by following the conversion shown in [77] (see proof of Theorem 2 of [77]). Now from [77], any single phase PRMT protocol over  $n = 2t_b + 1$  wires has to communicate  $\Omega\left(\frac{n\ell}{n-2t_b}\right)$  field elements. So any two phase PSMT protocol has to communicate  $\Omega\left(\frac{n\ell}{n-2t_b}\right)$  field elements as well.  $\square$

We next show that our two phase PSMT protocol 2-Optimal-PSMT-Static-Byzantine-Directed asymptotically satisfies the bound given in Theorem 13.7. Thus, we show that

protocol 2-Optimal-PSMT-Static-Byzantine-Directed is a communication optimal PSMT protocol.

**Theorem 13.8** *Protocol 2-Optimal-PSMT-Static-Byzantine-Directed is a communication optimal PSMT protocol and asymptotically satisfies the bound given in Theorem 13.7.*

PROOF: In protocol 2-Optimal-PSMT-Static-Byzantine-Directed, the communication complexity of **Phase I** is  $u$ . On the other hand, the communication complexity of **Phase II** is  $n(\delta + 1 - t_b) + n$ . Thus the communication complexity of protocol 2-Optimal-PSMT-Static-Byzantine-Directed is  $\mathcal{O}(n(\delta + 1 - t_b) + N)$  to securely send a message  $m^{\mathbf{S}}$  containing  $(\delta + 1 - t_b)$  field elements.

If  $u \leq t_b$ , then  $\delta = t_b$  and so protocol 2-Optimal-PSMT-Static-Byzantine-Directed securely sends a message  $m^{\mathbf{S}}$  containing one field element by communicating  $\mathcal{O}(N)$  field elements, where  $N = 3t_b + 1$ . From Theorem 13.7, if  $u \leq t_b$  and  $N = 3t_b + 1$ , then any two phase PSMT protocol must communicate  $\Omega(N)$  field elements to securely send a message containing single field element. This implies that protocol 2-Optimal-PSMT-Static-Byzantine-Directed is communication optimal, when  $0 < u \leq t_b$ .

If  $u > t_b$ , then  $\delta = u$  and so protocol 2-Optimal-PSMT-Static-Byzantine-Directed securely sends a message  $m^{\mathbf{S}}$  containing  $(u + 1) - t_b$  field elements by communicating  $\mathcal{O}(n(u + 1 - t_b))$  field elements, where  $n = 2t_b + 1$ . From Theorem 13.7, if  $u > t_b$  and  $n = 2t_b + 1$ , then any two phase PSMT protocol must communicate  $\Omega(n(u + 1 - t_b))$  field elements to securely send a message containing  $(u + 1 - t_b)$  field elements. This implies that protocol 2-Optimal-PSMT-Static-Byzantine-Directed is communication optimal, when  $u > t_b$ .  $\square$

We now move our discussion towards multiphase (more than two phase) PSMT protocols in directed networks tolerating  $\mathcal{A}_{t_b}^{static}$ . Before that, we briefly recall the guessing attack proposed in [88] against the PSMT protocol of [24, 62] in directed networks.

## 13.6 Guessing Attack Against PSMT Protocol of [24, 62] in Directed Networks

In [88], Yang et al. presented guessing attack against the PSMT protocols of [24, 62] in directed networks tolerating  $\mathcal{A}_{t_b}^{static}$ , which allows the adversary to gain extra information about the secret message. In this section, we briefly discuss this attack. We then show that our proposed two phase PSMT protocol 2-Optimal-PSMT-Static-Byzantine-Directed is secure against guessing attack.

We now present a *Guessing Attack* that takes advantage of how the feedback channels (i.e., the wires in the bottom band) are normally used. The current description of the attack is taken from [88]. In most protocols that work on networks with feedback channels, the feedbacks are used by the receiver **R** to seek for help from **S**, when **R** does not have enough information to recover the message (i.e., for reliability purpose). In guessing attack, the adversary does the following: since the adversary can choose to corrupt some feedback paths, it can simulate how **R** uses the feedback channels and learn from **S** the information it needs to recover the message with *better probability than guessing*. This allows the adversary to breach perfect privacy, as we describe now in more detail.

Here we give an example of how Guessing Attack breaches perfect secrecy of one of Desmedt and Wang's PSMT protocols given in [24]. We call the protocol as **DW**

protocol. A similar attack can be mounted on their other PSMT protocols. The current PSMT protocol corresponds to [24, Theorem 5]. First we shall sketch the **DW** protocol and then we show that it does not provide perfect secrecy.

1. **Condition for the DW protocol:** there are  $3t_b \geq 2t_b + 1$  wires in top band and one wire in bottom band <sup>1</sup>.
2. **Sketch of the DW protocol:** Let  $f_1, \dots, f_{3t_b}$  be the wires in the top band and  $b_1$  be the wire in the bottom band.
  - (a) ...
  - (b) **S** chooses a  $key^{\mathbf{S}} \in_R \mathbb{F}$ . **S** selects a random  $t_b$  degree polynomial  $f^{\mathbf{S}}(x)$  over  $\mathbb{F}$ , such that  $f^{\mathbf{S}}(0) = key^{\mathbf{S}}$  and computes the shares  $v = (s_1, \dots, s_{3t_b})$  of  $key^{\mathbf{S}}$ . Here  $s_i = f^{\mathbf{S}}(i)$ , for  $i = 1, \dots, 3t_b$ . **S** sends  $s_i$  to **R** via wire  $f_i$ .
  - (c) Let  $v^{\mathbf{R}} = (s_1^{\mathbf{R}}, \dots, s_{3t_b}^{\mathbf{R}})$  be the shares that **R** receive. If **R** finds that there are at most  $t_b - 1$  errors (using error-correcting code), **R** recovers  $key^{\mathbf{R}}$  from the shares, sends ‘stop’ to **S** via wire  $b_1$ ; otherwise, **R** sends  $v^{\mathbf{R}}$  to **S** via wire  $b_1$ .
  - (d) If **S** receives  $v^{\mathbf{S}} = (s_1^{\mathbf{S}}, \dots, s_{3t_b}^{\mathbf{S}})$  from wire  $b_1$ , then **S** broadcasts  $P = \{i : s_i^{\mathbf{S}} \neq s_i\}$  ( $|P| = t_b$ ) via top band; otherwise, **S** broadcasts ‘stop’.
  - (e) ...
  - (f) **S** broadcasts  $key^{\mathbf{S}} + m^{\mathbf{S}}$  via top band, where  $m^{\mathbf{S}}$  is the secret message.
  - (g) ...

The above single feedback channel protocol is the basis of the main PSMT protocols in [24]. We observe that this **DW** protocol is perfectly reliable, so in the above sketch we did not describe how **R** recovers the message (see [24] for the entire protocol). Now we show that using a Guessing Attack, the adversary can learn the message  $m^{\mathbf{S}}$  with probability better than guessing.

**Theorem 13.9** ([88]) *The above DW protocol is not a PSMT protocol from **S** to **R**.*

PROOF: Due to the fact that  $key^{\mathbf{S}} \in_R \mathbb{F}$ , if the **DW** protocol is perfectly secure, then the probability that the adversary guesses  $key^{\mathbf{S}}$  is  $\frac{1}{|\mathbb{F}|}$ . That is,  $\mathcal{A}_{t_b}^{static}$  learns nothing from the shares it gets, and can only guess a uniformly random number  $key^X \in \mathbb{F}$ , and with probability  $\frac{1}{|\mathbb{F}|}$ ,  $key^X = key^{\mathbf{S}}$ . We call this a *random guess*. Now we show a Guessing Attack by which the adversary can learn  $key^{\mathbf{S}}$  with a probability better than  $\frac{1}{|\mathbb{F}|}$ . The formal steps of the attack are given in Fig. 13.2.

In this Guessing Attack,  $\mathcal{A}_{t_b}^{static}$  guesses a share  $s_{t_b}^X$  and two keys  $key_1^X$  and  $key_2^X$ . It is straightforward that **S** will broadcast  $P$  if and only if **S** finds exactly  $t_b$  errors in  $v^X$ , and the  $t_b$  errors can only be either  $(s_{t_b+1}^X, \dots, s_{2t_b}^X)$  or  $(s_{2t_b+1}^X, \dots, s_{3t_b}^X)$ . That is, the guess is successful if  $s_{t_b}^X = s_{t_b}$  and one of the two keys is correct (i.e.,  $key_i^X = key^{\mathbf{S}}, i \in \{1, 2\}$ ). Thus the probability  $T$  that the guess is successful is

$$T = \frac{1}{|\mathbb{F}|} \times \left( 2 \times \frac{1}{|\mathbb{F}|} \right) = \frac{2}{|\mathbb{F}|^2}.$$

---

<sup>1</sup>This condition is sufficient for PSMT, but is stronger than the necessary condition. See [24] for more details.

Figure 13.2: Guessing Attack on **DW** Protocol

**DW-Guessing Attack**

$\mathcal{A}_{t_b}^{static}$  chooses to control wires  $f_1, \dots, f_{t_b-1}$  and wire  $b_1$  from top band and bottom band respectively. Thus  $\mathcal{A}_{t_b}^{static}$  is able to get the shares  $(s_1, \dots, s_{t_b-1})$  in Step 2b of **DW** protocol. With these  $t_b - 1$  shares,  $\mathcal{A}_{t_b}^{static}$  does the following:

1.  $\mathcal{A}_{t_b}^{static}$  chooses a share  $s_{t_b}^X \in_R \mathbb{F}$  and two keys  $key_1^X, key_2^X \in_R \mathbb{F}$  ( $key_1^X \neq key_2^X$ ).
2. Corresponding to  $key_1^X$ , the adversary assumes that  $(s_1, \dots, s_{t_b-1}, s_{t_b}^X)$  are  $t_b$  shares of  $key_1^X$  and thus using Lagrange interpolation,  $\mathcal{A}_{t_b}^{static}$  gets another  $t_b$  shares  $(s_{2t_b+1}^X, \dots, s_{3t_b}^X)$  of  $key_1^X$ .
3. Similarly, corresponding to  $key_2^X$ , the adversary assumes that  $(s_1, \dots, s_{t_b-1}, s_{t_b}^X)$  are  $t_b$  shares of  $key_2^X$  and thus using Lagrange interpolation,  $\mathcal{A}_{t_b}^{static}$  gets another  $t_b$  shares  $(s_{2t_b+1}^X, \dots, s_{3t_b}^X)$  of  $key_2^X$ .
4.  $\mathcal{A}_{t_b}^{static}$  sets  $v^X = (s_1, \dots, s_{t_b-1}, s_{t_b}^X, \dots, s_{3t_b}^X)$ .
5. In each execution step of the **DW** protocol,  $\mathcal{A}_{t_b}^{static}$  acts passively on wires  $f_1, \dots, f_{t_b-1}$ . Thus **R** sends ‘stop’ to **S** in Step 2c of **DW** protocol.
6. On the feedback path  $b_1$  that is under the control of  $\mathcal{A}_{t_b}^{static}$ , the adversary ignores what **R** sends and forwards  $v^X$  to **S**.
7. Then in Step 2d of **DW** protocol, if **S** finds exactly  $t_b$  errors in  $v^S = v^X$ , then **S** broadcasts  $P = \{i : s_i^X \neq s_i\}$ , according to which  $\mathcal{A}_{t_b}^{static}$  recovers  $key^S = key_j^X$  ( $j \in \{1, 2\}$ ); otherwise, **S** broadcasts ‘stop’ and  $\mathcal{A}_{t_b}^{static}$  randomly guesses a  $key^X$ .

If the guess fails, then  $\mathcal{A}_{t_b}^{static}$  will use a random guess with probability  $\frac{1}{|\mathbb{F}|}$  to get  $key^X = key^S$ . Thus, the total probability  $G$  that  $\mathcal{A}_{t_b}^{static}$  learns  $key^S$  by performing Guessing Attack is

$$G = T + (1 - T) \times \frac{1}{|\mathbb{F}|} > \frac{1}{|\mathbb{F}|}.$$

Therefore,  $\mathcal{A}_{t_b}^{static}$  can learn  $key^S$  with a probability better than  $\frac{1}{|\mathbb{F}|}$  and simultaneously recover  $m^S$  with probability better than guessing. Hence the **DW** protocol is not perfectly secure.  $\square$

Note that in [87], Wang and Desmedt provided a new PSMT protocol that uses induction when **S** receives tuples of shares in feedbacks. When **S** notices that Guessing Attack may happen according to the feedbacks it receives, it uses an induction and re-sends the message without revealing the message to the adversary (0-private). The property of the threshold adversary allows the induction to be continued until the message is transmitted 0-reliably. Thus the protocol in [87] enables perfect security. For details of the PSMT protocol tolerating a threshold adversary, we refer to [87, Theorem 4.2].

As we have seen above, the basic idea of Guessing Attack is to replace the feedbacks

from  $\mathbf{R}$  to  $\mathbf{S}$  on the feedback channel with something that may reveal the message. There is some probability associated with this guessing of being successful.

Besides the Desmedt-Wang protocols, Yang et al. [88] observed that all the PSMT protocols in directed networks against  $\mathcal{A}_{t_b}^{static}$  as given in [62] do not guarantee perfect privacy when the Guessing Attack takes place. For complete details, see [88].

### 13.6.1 Security of Two Phase PSMT Protocol Against Guessing Attack

In the guessing attack, the feedback sent by  $\mathbf{R}$  is dependent on the information which  $\mathbf{S}$  has sent to it, which is further dependent on the secret message. So the adversary can modify the feedback (without letting  $\mathbf{S}$  know about it) and wait for  $\mathbf{S}$ 's response regarding the feedback. From  $\mathbf{S}$ 's response, the adversary will come to know about the status of the incorrect feedback that he has sent to  $\mathbf{S}$ . This further allows the adversary to get extra information about the message.

In protocol 2-Optimal-PSMT-Static-Byzantine-Directed, the information sent by  $\mathbf{R}$  is not any feedback information and it is a complete random information. So unlike **DW** protocol,  $\mathbf{S}$  does not have to make any response about the received information. Rather,  $\mathbf{S}$  assumes the received information to be a genuine information and uses it to construct the one time pad, about which the adversary will have no extra information, other than guessing. In this way, protocol 2-Optimal-PSMT-Static-Byzantine-Directed is secure against guessing attack.

Similarly, all the PSMT protocols in *undirected* networks presented in earlier chapters of this thesis are secure against guessing attack. This is because in all these protocols,  $\mathbf{R}$  reliably sends back the feedback by broadcasting. So  $\mathbf{S}$  always correctly receives the feedback, as all the wires are bi-directional and there are sufficient number of wires for reliable communication.

The vulnerability of the PSMT protocols of [24] and [62] against guessing attack bring forth the issue of designing efficient PSMT protocols in directed networks tolerating  $\mathcal{A}_{t_b}^{static}$ , which are secure against guessing attack. We proceed to do so in the next section.

## 13.7 New Multi Phase PSMT Protocols Secure Against Guessing Attack

In this section, we propose new multiphase (more than two phase) PSMT protocols in directed networks, tolerating  $\mathcal{A}_{t_b}^{static}$ , which are secure against guessing attack. We also show that the communication complexity of our proposed protocols are asymptotically tight. For this, we derive non trivial lower bound on the communication complexity of multiphase PSMT protocols in directed networks. Our PSMT protocols satisfy the characterization of PSMT in directed networks given in [24]. Thus our protocols require  $u > 0$  wires in the bottom band and  $n = \max(3t_b - 2u + 1, 2t_b + 1)$  wires in the top band.

### 13.7.1 Three Phase PSMT with $0 < u \leq \frac{t_b}{2}$ and $n = 3t_b - 2u + 1$

In this section, we present a three phase PSMT protocol called 3-Optimal-PSMT-Static-Byzantine-Directed, which securely sends a message  $m^{\mathbf{S}}$  containing  $\ell = n^2u$  field elements by communicating  $\mathcal{O}(n^3u) = \mathcal{O}(n\ell)$  field elements.

Informally the protocol works as follows: **S** tries to correctly establish an information theoretically secure one time pad of size  $n^2u$  with **R**. Let  $\mathcal{C}$  denote the set of all  $RS$  codewords of length  $n = 3t_b - 2u + 1$  over  $\mathbb{F}$ , encoded using all possible polynomials of degree  $t_b$  over  $\mathbb{F}$ , for fixed  $\alpha_1, \dots, \alpha_n$ . Here  $\alpha_i$  is associated with wire  $f_i$ . Hence the hamming distance between any two codewords in  $\mathcal{C}$  is  $n - t_b = 2t_b - 2u + 1 \geq t_b + 1$ . In protocol 3-Optimal-PSMT-Static-Byzantine-Directed, **S** selects a number of random codewords from  $\mathcal{C}$  and sends them across the  $n$  wires. **R** receives the vectors and finds the pseudo-basis of the received vectors. **R** will be able to do so, as the hamming distance between any two codewords in  $\mathcal{C}$  is at least  $t_b + 1$ . **R** then sends the pseudo-basis, pseudo-dimension and index set through the *bottom band*.

We say that a pseudo-basis, pseudo-dimension and index set triple received over a wire in bottom band is **valid** iff all the vectors listed in pseudo-basis differs from the corresponding original codewords (sent by **S**) in at most  $t_b$  locations. Note that **S** has no knowledge on whether the original pseudo-basis generated by **R** is received by him. This is because there may not be enough number of honest wires in the bottom band to reliably receive the pseudo-basis generated by **R**. So **S** broadcasts all the valid triple of (pseudo basis, pseudo-dimension and index set) as received by him along with the corresponding list of corrupted wires. Now **R** correctly receives all the pseudo-basis, pseudo-dimension and index set (as received by **S**), along with their corresponding list of corrupted wires.

**R** checks whether the pseudo-basis generated by him is present in the received list of pseudo-basis. If yes then he knows the set of corrupted wires and can recover all the original codewords (sent by **S**) by neglecting the values received over those corrupted wires during first phase. Otherwise **R** learns that entire *bottom band* is corrupted and hence in the *top band* there are at most  $t_b - u$  Byzantine faults. So **R** can correct these  $t_b - u$  errors in each vector, received during first phase and thus can recover all the original codewords. Hence in any case **S** and **R** will agree on all the codewords chosen by **S**.

The above steps are enough to ensure perfect reliability. But during the transmission of pseudo-basis over  $u$  wires,  $\mathcal{A}_{t_b}^{static}$  can generate  $u$  distinct **valid** pseudo-basis each containing at most  $t_b$  disjoint vectors (this he can do by guessing with non-zero probability, as in the case of guessing attack). This will make the protocol vulnerable against guessing attack. To deal with this situation, initially **S** should send sufficient number of codewords such that after removing all the  $ut_b$  codewords corresponding to the vectors appearing in the received list of valid pseudo-basis, the remaining codewords can be used to construct an information theoretic secure pad of size  $n^2u$ . Once the pad is established, **S** uses the pad to blind the message and sends the blinded message reliably to **R**. The protocol is now formally given in Fig. 13.3.

We now formally prove the properties of protocol 3-Optimal-PSMT-Static-Byzantine-Directed.

**Theorem 13.10 (Perfect Reliability)** *Protocol 3-Optimal-PSMT-Static-Byzantine-Directed provides perfect reliability.*

PROOF: First note that since  $n = 3t_b - 2u + 1 \geq 2t_b + 1$ , any information broadcast by **S** over the top band will be received by **R** correctly. This implies that **R** correctly receives blinded message  $\Gamma$  and either one of the following two (depending upon what **S** has broadcasted during **Phase III**): all quadruples  $(\mathcal{B}^j, p^j, \mathcal{I}^j, CORRUPTED^j)$  or the message “**Entire Bottom band is corrupted**”. Now to prove that **R** correctly recovers the message  $m^S$  sent by **S**, we show that **S** and **R** shares the same pad  $Z$ . **S** and **R** will share  $Z$  if

1.  $\Lambda$  is same at both ends and
2.  $\mathbf{R}$  is able to recover all the polynomials  $F_i(x)$  for  $i \notin \Lambda$ .

Since  $\mathbf{S}$  broadcasts all valid triples to  $\mathbf{R}$ ,  $\Lambda$  will be same at both ends. Now we show that irrespective of the behavior of  $\mathcal{A}_{t_b}^{static}$ ,  $\mathbf{R}$  will always recover all the polynomials  $F_i(x)$  for  $i \notin \Lambda$ .

If  $\mathcal{A}_{t_b}^{static}$  spares (either does not control or behave passively) at least one wire, say  $b_j$ , in the *bottom band*, then  $\mathbf{S}$  will correctly receive  $(\mathcal{B}^j, p^j, \mathcal{I}^j) = (\mathcal{B}, p, \mathcal{I})$  and hence  $CORRUPTED^j$  will contain all the wires which were corrupted during first phase (see the properties of pseudo-basis). In this case,  $\mathbf{R}$  will correctly receive  $CORRUPTED^j$ , from which it identifies all wires (at most  $t_b$ ) which were corrupted during first phase.  $\mathbf{R}$  ignores the values received over those wires during **Phase I** and with the remaining values (at least  $t_b + 1$ ) all the polynomials can be recovered correctly.

On the other hand, if  $\mathcal{A}_{t_b}^{static}$  corrupts the entire bottom band such that either  $\mathbf{S}$  detects that all the received triples are invalid or  $\mathbf{R}$  detects that his original triple is not present in the list of triples received by  $\mathbf{S}$  (at the end of **Phase II**), then  $\mathbf{R}$  concludes that entire *bottom band* is corrupted. Hence  $\mathbf{R}$  applies RS decoding on the received vector  $Y_i$ 's to correct  $t_b - u$  errors (see Theorem 2.19) and reconstruct polynomial  $F_i(x)$  for  $i \notin \Lambda$ . Hence the theorem.  $\square$

**Theorem 13.11 (Perfect Security)** *In Protocol 3-Optimal-PSMT-Static-Byzantine-Directed,  $m^{\mathbf{S}}$  will be information theoretically secure.*

PROOF: The message  $m^{\mathbf{S}}$  will be information theoretically secure from  $\mathcal{A}_{t_b}^{static}$  if the pad  $Z$  is information theoretically secure. According to the protocol,  $Z$  contains  $F_i(0)$  iff  $i \notin \Lambda$ . Notice that  $\Lambda = \cup_j \{\mathcal{I}^j | (\mathcal{B}^j, p^j, \mathcal{I}^j) \text{ is a valid triple}\}$ . Now a valid triple  $(\mathcal{B}^j, p^j, \mathcal{I}^j)$  can be either the original triple  $(\mathcal{B}, p, \mathcal{I})$  sent by  $\mathbf{R}$  or it may be different from  $(\mathcal{B}, p, \mathcal{I})$  and generated by  $\mathcal{A}_{t_b}^{static}$  (who can guess it with non-zero probability). In the former case  $(\mathcal{B}^j, p^j, \mathcal{I}^j)$  may be eavesdropped by  $\mathcal{A}_{t_b}^{static}$  during its transmission over the *bottom band*. In later case,  $\mathcal{A}_{t_b}^{static}$  knows  $(\mathcal{B}^j, p^j, \mathcal{I}^j)$  since he himself has generated them. Hence it is possible that all  $F_i(0)$ 's with  $i \in \Lambda$  are already exposed to  $\mathcal{A}_{t_b}^{static}$ . But for remaining polynomials,  $\mathcal{A}_{t_b}^{static}$  knows at most  $t_b$  points on them (by listening during first phase) and hence constant term of each  $F_i(x)$  with  $i \notin \Lambda$  will be information theoretically secure.  $\square$

**Theorem 13.12** *Protocol 3-Optimal-PSMT-Static-Byzantine-Directed sends a message  $m^{\mathbf{S}}$  containing  $\ell = n^2u$  field elements by communicating  $\mathcal{O}(n^3u) = \mathcal{O}\left(\frac{n\ell}{n-(3t_b-2u)}\right) = \mathcal{O}(n\ell)$  field elements.*

PROOF: During **Phase I**,  $\mathbf{S}$  communicates  $P = n^2u + ut_b$  codewords to  $\mathbf{R}$  which requires communication complexity of  $Pn = n^3u + nut_b = \mathcal{O}(n^3u)$  field elements. In **Phase II**,  $\mathbf{R}$  sends the triple  $(\mathcal{B}, p, \mathcal{I})$  through the *bottom band*. This incurs a communication cost of  $\mathcal{O}(nt_b.u + 1.u + t_b.u) = \mathcal{O}(n^2u)$ . In the worst case, it may happen that over every wire in *bottom band*,  $\mathbf{S}$  receives a distinct valid triple  $(\mathcal{B}^j, p^j, \mathcal{I}^j)$ . Then communication complexity of **Phase III** for sending the triples will be  $\mathcal{O}(n^2u.n) = \mathcal{O}(n^3u)$ . Since message is of size  $n^2u$ , broadcasting the blinded message  $\Gamma$  results in a communication cost of  $\mathcal{O}(n^3u)$ . Hence overall communication complexity of protocol 3-Optimal-PSMT-Static-Byzantine-Directed is  $\mathcal{O}(n^3u)$ .  $\square$

### 13.7.2 Six Phase PSMT when $\frac{t_b}{2} < u \leq t_b$ and $n = 2t_b + 1$

If  $\frac{t_b}{2} < u \leq t_b$ , then  $n = \max(3t_b - 2u + 1, 2t_b + 1) = 2t_b + 1$ . In this section, we present a six phase PSMT protocol called 6-Optimal-PSMT-Static-Byzantine-Directed where  $n = 2t_b + 1$  and  $\frac{t_b}{2} < u \leq t_b$ . Protocol 6-Optimal-PSMT-Static-Byzantine-Directed securely sends a message  $m^{\mathbf{S}}$  containing  $\ell = n^2u$  field elements by communicating  $\mathcal{O}\left(\frac{n\ell}{n-(3t_b-2u)}\right) = \mathcal{O}\left(\frac{n^3u}{2u-t_b}\right) = \mathcal{O}\left(\frac{n\ell}{2u-t_b+1}\right)$  field elements. Interestingly, when  $u = \frac{t_b}{2} + \Theta(t_b)$ , then Protocol 6-Optimal-PSMT-Static-Byzantine-Directed securely sends  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements.

Protocol 6-Optimal-PSMT-Static-Byzantine-Directed achieves its goal by allowing  $\mathbf{S}$  and  $\mathbf{R}$  to share  $\frac{n^2u}{2u-t_b+1}$  common polynomials each of degree  $2u$ , such that  $\mathcal{A}_{t_b}^{static}$  knows only  $t_b$  points on each of them. Once this is done, both  $\mathbf{S}$  and  $\mathbf{R}$  can generate an information theoretic pad of length  $n^2u$  by using EXTRAND algorithm.  $\mathbf{S}$  can then blind the message and sends it to  $\mathbf{R}$ . However, note that  $\mathbf{S}$  cannot send the blinded message to  $\mathbf{R}$  by broadcasting it over the entire *top* band, as done in protocol 3-Optimal-PSMT-Static-Byzantine-Directed. Because the communication complexity will then become  $\mathcal{O}(n^3u)$ . So  $\mathbf{S}$  reliably sends the blinded message by using the protocol 3-Optimal-PRMT-Static-Byzantine-Directed given in Fig. 7.2, which takes three phases. Since here  $n = 2t_b + 1$  and  $(n - 2t_b) + 2u = \Theta(n)$ , we can execute 3-Optimal-PRMT-Static-Byzantine-Directed.  $\mathbf{R}$  can recover the message since he knows the pad.

In 6-Optimal-PSMT-Static-Byzantine-Directed,  $\mathcal{C}$  denotes the set of all possible RS codewords of length  $N = n + u = 2t_b + 1 + u$  encoded using all possible polynomials of degree  $2u > t_b$  over  $\mathbb{F}$ . Hence the hamming distance between any two codeword in  $\mathcal{C}$  is  $N - 2u = 2t_b - u + 1 \geq t_b + 1$ . Protocol 6-Optimal-PSMT-Static-Byzantine-Directed is formally given in Fig. 13.4.

We now prove the properties of protocol 6-Optimal-PSMT-Static-Byzantine-Directed.

**Theorem 13.13 (Perfect Reliability)** *In protocol 6-Optimal-PSMT-Static-Byzantine-Directed,  $\mathbf{R}$  correctly recovers  $m^{\mathbf{S}}$ .*

PROOF: First note that for each codeword  $C_i$ , the corresponding vector  $Y_i$  of length  $N$ , possessed by  $\mathbf{R}$ , differs from  $C_i$  only at  $t_b$  locations. This is because  $\mathcal{A}_{t_b}^{static}$  controls at most  $t_b$  wires, including top band and bottom band. With this observation, the correctness proof of this theorem simply follows from the correctness proof of protocol 3-Optimal-PSMT-Static-Byzantine-Directed, protocol 3-Optimal-PRMT-Static-Byzantine-Directed and algorithm EXTRAND.  $\square$

**Theorem 13.14 (Perfect Security)** *In Protocol 6-Optimal-PSMT-Static-Byzantine-Directed,  $m^{\mathbf{S}}$  will be information theoretically secure.*

PROOF: The secrecy of the message follows using similar argument as in Theorem 13.11 and the properties of EXTRAND algorithm. Notice that in the protocol, each codeword  $C_i$ , such that  $i \notin \Lambda$  corresponds to a  $2u$  degree polynomial and adversary will know at most  $t_b$  points on the polynomial. This implies that  $2u + 1 - t_b$  coefficients of the polynomial and hence  $2u + 1 - t_b$  components of  $C_i$  will be information theoretically secure. This is why  $\mathbf{S}$  extracts a random pad  $Z^i$  of size  $2u + 1 - t_b$  from each  $C_i$ , such that  $i \notin \Lambda$ .  $\square$

**Theorem 13.15** *Protocol 6-Optimal-PSMT-Static-Byzantine-Directed securely sends a message  $m^{\mathbf{S}}$  containing  $\ell = n^2u$  field elements by communicating  $\mathcal{O}\left(\frac{n\ell}{n-(3t_b-2u)}\right) = \mathcal{O}\left(\frac{n\ell}{2u-t_b+1}\right)$  field elements.*

PROOF: During **Phase I**, **R** sends  $Q = \frac{n^2u}{2u-t_b+1} + ut_b$  vectors, each of size  $u$ , thus communicating  $Qu = \mathcal{O}(\frac{n^2u^2}{2u-t_b+1} + u^2t_b)$  field elements. During **Phase II**, **S** communicates  $Q = \frac{n^2u}{2u-t_b+1} + ut_b$  codewords to **R** which incurs a communication cost of  $Qn = \frac{n^3u}{2u-t_b+1} + nut_b$  field elements. In **Phase III**, **R** sends the triple  $(\mathcal{B}, p, \mathcal{I})$  through the bottom band. This incurs a communication cost of  $\mathcal{O}(nt_b.u + 1.u + t_b.u) = \mathcal{O}(n^2u)$ . In worst case it may happen that over every wire in *bottom band*, **S** receives a distinct valid triple  $(\mathcal{B}^j, p^j, \mathcal{I}^j)$ . Then communication complexity of **Phase IV** for sending the triples using protocol 3-Optimal-PRMT-Static-Byzantine-Directed will be  $\mathcal{O}(n^2u)$ . Similarly sending the blinded message  $\Gamma$  of size  $n^2u$  using protocol 3-Optimal-PRMT-Static-Byzantine-Directed results in a communication cost of  $\mathcal{O}(n^2u)$ . Hence overall communication complexity of protocol 6-Optimal-PSMT-Static-Byzantine-Directed is  $\mathcal{O}(\frac{n^3u}{2u-t_b+1})$ .  $\square$

### 13.7.3 Six Phase PSMT Protocol when $u > t_b$ and $n = 2t_b + 1$

If  $u > t_b$ , then from [24], we require  $n = \max(3t_b - 2u + 1, 2t_b + 1) = 2t_b + 1$  wires in the top band for the existence of any multiphase (more than two phase) PSMT protocol. If  $u = t_b$  and  $n = 2t_b + 1 = \Theta(t_b)$ , then from Theorem 13.15, protocol 6-Optimal-PSMT-Static-Byzantine-Directed securely sends  $\ell = n^2u = \Theta(n^3)$  field elements by communicating  $\mathcal{O}(n^3)$  field elements. Hence, if  $u > t_b$  and  $n \geq 2t_b + 1$ , then **S** and **R** can execute 6-Optimal-PSMT-Static-Byzantine-Directed by considering the first  $2t_b + 1$  wires in the top band and first  $t_b$  wires in the bottom band. Thus, we have the following theorem:

**Theorem 13.16** *Suppose  $n \geq 2t_b + 1$  and  $u > t_b$ . Then there exists a six phase PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , which securely sends  $\ell$  ( $\ell = \Theta(n^3)$ ) field elements by communicating  $\mathcal{O}(\ell)$  field elements.*

We now show that protocol 3-Optimal-PSMT-Static-Byzantine-Directed and 6-Optimal-PSMT-Static-Byzantine-Directed are communication optimal. For this, we derive non-trivial lower bound on the communication complexity of multiphase PSMT protocols in directed networks tolerating  $\mathcal{A}_{t_b}^{static}$ .

## 13.8 Lower Bound on Communication Complexity of Multi Phase PSMT

Recall that from Theorem 13.2, any three or more phase PSMT protocol requires  $n = \max(3t_b - 2u + 1, 2t_b + 1)$  wires in the top band to tolerate  $\mathcal{A}_{t_b}^{static}$ . To build our lower bound argument for three or more phase PSMT protocol, we need the concept of secret sharing and Maximum Distance Separable (MDS) codes [45].

**Definition 13.17** ( *$x$ -out-of- $n$  Secret Sharing Scheme (SSS) [71]*) *An  $x$ -out-of- $n$  Secret Sharing Scheme (SSS) is a probabilistic function  $S : \mathbb{F} \rightarrow \mathbb{F}^n$  with the property that for any  $M \in \mathbb{F}$  and  $S(M) = (s_1, \dots, s_n)$ , no information on  $M$  can be inferred from any  $x$  elements of  $(s_1, \dots, s_n)$  and  $M$  can be recovered from any  $x + 1$  elements in  $(s_1, \dots, s_n)$ .*

The set of all possible  $(s_1, \dots, s_n)$  can be viewed as a code and its elements as codewords [24]. If the code is a Maximum Distance Separable (MDS) code [45, 24] (e.g RS code), then it can correct  $c$  errors and simultaneously detect  $d$  additional errors

iff  $n - x > 2c + d$  [45, 24]. An  $x$ -out-of- $n$  SSS is called MDS  $x$ -out-of- $n$  SSS if it is constructed from a MDS code. MDS SSSs can be constructed from any MDS codes, for example RS codes [45, 46, 24]. So we have the following theorem on the error correction and detection capability of MDS  $x$ -out-of- $n$  SSS:

**Theorem 13.18** ([45, 24]) *Any MDS  $x$ -out-of- $n$  SSS can correct  $c$  errors and detect  $d$  additional errors in a codeword iff  $n - x > 2c + d$ .*

We next give the following definition:

**Definition 13.19** ( $(\alpha, \beta, \gamma, m, \ell)$ -SSS) *Given a secret  $m$  containing  $\ell$  field elements from  $\mathbb{F}$ , an  $(\alpha, \beta, \gamma, m, \ell)$ -SSS generates  $\alpha$  shares of  $m$ , such that any set of  $\beta$  shares has full information about the secret  $m$ , while any set of  $\gamma$  shares has no information about the secret  $m$ , where  $\alpha > \beta > \gamma$ .*

Now the lower bound on the communication complexity of multiphase PSMT in directed network is given by the following theorem:

**Theorem 13.20** *Suppose there exists  $u$  wires in the bottom band and  $n \geq \max(3t_b - 2u + 1, 2t_b + 1)$  wires in the top band. Then any three or more phase PSMT protocol that securely sends a message  $m$  containing  $\ell$  field elements from  $\mathbb{F}$  tolerating  $\mathcal{A}_{t_b}^{static}$  must communicate:*

1.  $\Omega(\frac{n\ell}{n-(3t_b-2u)})$  field elements when  $0 < u \leq t_b$ ;
2.  $\Omega(\ell)$  field elements when  $u > t_b$ .

PROOF: We first prove part 1 of the theorem. The outline of the proof strategy is as follows: we first show that the communication complexity of any three or more phase PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  to send a message  $m \in \mathbb{F}^\ell$  is not less than the share complexity (sum of all the shares) of an  $(n, (n - 2(t_b - u)), t_b, m, \ell)$ -SSS (see Lemma 13.21). We then show that the share complexity of any  $(n, (n - 2(t_b - u)), t_b, m, \ell)$ -SSS is  $\Omega(\frac{n\ell}{n-(3t_b-2u)})$  field elements (see Lemma 13.22). Part 1 of Theorem 13.20 will then follow from Lemma 13.21 and Lemma 13.22. So we now proceed to prove Lemma 13.21.

**Lemma 13.21** *Let  $0 < u \leq t_b$  and  $n = \max(3t_b - 2u + 1, 2t_b + 1)$ . Then the communication complexity of any three or more phase PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  to send a message  $m \in \mathbb{F}^\ell$  is not less than the share complexity (sum of all the shares) of an  $(n, (n - 2(t_b - u)), t_b, m, \ell)$ -SSS.*

PROOF: Let  $\Pi$  be a PSMT protocol which runs for  $p \geq 3$  phases. Without loss of generality, let the view of  $\mathbf{S}$  in  $\Pi$ , denoted by  $view_{\Pi}^{\mathbf{S}}$  be drawn from a probability distribution that depends on the message  $m$ , the coin flips  $\mathcal{R}^{\mathbf{S}}$  of  $\mathbf{S}$ , the coin flips  $\mathcal{R}^{\mathbf{R}}$  of  $\mathbf{R}$  and the coin flips  $\mathcal{R}^{\mathcal{A}}$  of  $\mathcal{A}_{t_b}^{static}$ . Without loss of generality, we assume that the value of  $\mathcal{R}^{\mathcal{A}}$  will determine the choice of faulty wires controlled by  $\mathcal{A}_{t_b}^{static}$ . Without loss of generality, we assume that  $\mathbf{S}$  is silent in even phases and  $\mathbf{R}$  is silent in odd phases [24]. Now consider the following possible strategy for  $\mathcal{A}_{t_b}^{static}$  in  $\Pi$ :

1. First  $\mathcal{A}_{t_b}^{static}$  uses  $\mathcal{R}^{\mathcal{A}}$  to choose a value  $r$ .
2. If  $r = 0$ , then  $\mathcal{A}_{t_b}^{static}$  uses  $\mathcal{R}^{\mathcal{A}}$  to choose  $t_b$  wires  $f_{j_1}, f_{j_2}, \dots, f_{j_{t_b}}$  from the top band and behaves passively over these paths. This means the adversary proceeds according to protocol  $\Pi$ .

3. If  $r = 1$ , then  $\mathcal{A}_{t_b}^{static}$  uses  $\mathcal{R}^A$  to choose  $t_b - u$  wires  $f_{j_1}, f_{j_2}, \dots, f_{j_{t_b-u}}$  from the *top* band and all the  $u$  wires from the *bottom* band. In this case  $\mathcal{A}_{t_b}^{static}$  corrupts all the  $u$  wires in the bottom band and the  $t_b - u$  wires  $f_{j_1}, f_{j_2}, \dots, f_{j_{t_b-u}}$  from the top band.  $\mathcal{A}_{t_b}^{static}$  also uses  $\mathcal{R}^A$  to choose a message  $\bar{m} \in \mathbb{F}$  according to the same probability distribution from which the actual message  $m$  was drawn. Now over the corrupted wires,  $\mathcal{A}_{t_b}^{static}$  behaves in the following way:

- (a) Over the wires  $f_{j_1}, f_{j_2}, \dots, f_{j_{t_b-u}}$ , it ignores what  $\mathbf{S}$  sends in odd phases of  $\Pi$  and simulates what  $\mathbf{S}$  would send to  $\mathbf{R}$  if  $\bar{m}$  would have been the message.
- (b) Over the paths in the bottom band, it ignores what  $\mathbf{R}$  sends to  $\mathbf{S}$  in even phases of  $\Pi$  and simulates what  $\mathbf{R}$  would send to  $\mathbf{S}$  when  $r = 0$ .

$\mathcal{A}_{t_b}^{static}$  can behave in the above manner with non-zero probability. Now let  $\alpha_{i,j}^{\mathbf{S}}$  be the values that  $\mathbf{S}$  sends on wire  $f_i$  in phase  $j$  of protocol  $\Pi$ . Let  $\alpha_i^{\mathbf{S}} = (\alpha_{i,1}^{\mathbf{S}}, \dots, \alpha_{i,p}^{\mathbf{S}})$  i.e.  $\alpha_i^{\mathbf{S}}$  is the concatenation of the values sent by  $\mathbf{S}$  over wire  $f_i$  during the execution of  $\Pi$ . We can view  $\alpha_i^{\mathbf{S}}$ 's as the shares of message  $m$ . Now if  $r = 0$ , due to the fact that  $\Pi$  is a PSMT protocol,  $\mathcal{A}_{t_b}^{static}$  should not get any information on  $m$  from any  $t_b$  shares from the set  $\{\alpha_1^{\mathbf{S}}, \dots, \alpha_n^{\mathbf{S}}\}$ . This implies that  $(\alpha_1^{\mathbf{S}}, \dots, \alpha_n^{\mathbf{S}})$  is an  $x$ -out-of- $n$  SSS for  $x \geq t_b$ . Note that when  $x > t_b$ , it is still ensured that  $t_b$  shares from the set  $\{\alpha_1^{\mathbf{S}}, \dots, \alpha_n^{\mathbf{S}}\}$  do not reveal any information on  $m$ . Now if  $r = 1$ , due to the fact that  $\Pi$  is also a PRMT protocol,  $\mathbf{R}$  must be able to correct any  $t_b - u$  errors in the shares  $(\alpha_1^{\mathbf{S}}, \dots, \alpha_n^{\mathbf{S}})$  and thus recover the message  $m$ . Summing up,  $(\alpha_1^{\mathbf{S}}, \dots, \alpha_n^{\mathbf{S}})$  is an MDS  $x$ -out-of- $n$  SSS with the capability of correcting  $t_b - u$  error where  $x \geq t_b$ . Now by Theorem 13.18, an MDS  $x$ -out-of- $n$  SSS can correct  $(t_b - u)$  errors if

$$n - x > 2(t_b - u) \Rightarrow x < n - 2(t_b - u) \Rightarrow x + 1 \leq n - 2(t_b - u). \quad (13.2)$$

This shows that the communication done by  $\mathbf{S}$  (alone) is equivalent to the share complexity (sum of all the shares) of an  $(n, (n - 2(t_b - u)), t_b, m, \ell)$ -SSS. Thus ignoring the communication done by  $\mathbf{R}$ , we can say that the communication done in protocol  $\Pi$  is not less than the share complexity (sum of all the shares) of an  $(n, (n - 2(t_b - u)), t_b, m, \ell)$ -SSS.  $\square$

**Lemma 13.22** *The share complexity of any  $(n, (n - 2(t_b - u)), t_b, m, \ell)$ -SSS is  $\Omega(\frac{n\ell}{n - (3t_b - 2u)})$  field elements.*

PROOF: We define the following notations:

1.  $\mathcal{M}$  denotes the message space from where the message  $m$  is selected. In our context,  $\mathcal{M} = \mathbb{F}^\ell$ .
2. For  $i = 1, \dots, n$ ,  $\mathbf{X}_i^m$  denotes the set of all possible  $i^{th}$  share corresponding to message  $m \in \mathcal{M}$ , that could be generated by any  $(n, (n - 2(t_b - u)), t_b, m, \ell)$ -SSS.
3. For  $j \geq i$ ,  $\mathbf{M}_{i,j}^m \subseteq \mathbf{X}_i^m \times \mathbf{X}_{i+1}^m \times \dots \times \mathbf{X}_j^m$  denotes the set of all possible  $\{i^{th}, (i + 1)^{th}, \dots, j^{th}\}$  shares, corresponding to message  $m \in \mathcal{M}$ , that could be generated by any  $(n, (n - 2(t_b - u)), t_b, m, \ell)$ -SSS.
4.  $\mathbf{M}_{i,j} = \bigcup_{m \in \mathcal{M}} \mathbf{M}_{i,j}^m$  and  $\mathbf{X}_i = \bigcup_{m \in \mathcal{M}} \mathbf{X}_i^m$ . We call  $\mathbf{X}_i$  as the *capacity* of  $i^{th}$  share and  $\mathbf{M}_{i,j}$  as the *capacity* of the set of  $\{i^{th}, (i + 1)^{th}, \dots, j^{th}\}$  shares.

To generate  $n$  shares for message  $m$ , any  $(n, (n - 2(t_b - u)), t_b, m, \ell)$ -SSS would select one element from the set  $\mathbf{X}_i$ , as the  $i^{\text{th}}$  share of  $m$ , for  $i = 1, \dots, n$ . Each element of the set  $\mathbf{X}_i$  can be represented by  $\log |\mathbf{X}_i|$  bits. Thus, the share complexity corresponding to message  $m$  will be  $\sum_{i=1}^n \log |\mathbf{X}_i|$  bits. In the sequel, we show that  $\sum_{i=1}^n \log |\mathbf{X}_i| \geq \left( \frac{n\ell \log(|\mathbb{F}|)}{n - (3t_b - 2u)} \right)$ .

From the property of a  $(n, (n - 2(t_b - u)), t_b, m, \ell)$ -SSS, any set of  $t_b$  shares is *independent* of the message. Thus, for any two messages  $m_1, m_2 \in \mathcal{M}$ , the following should hold:

$$\mathbf{M}_{2t_b - 2u + 1, 3t_b - 2u}^{m_1} = \mathbf{M}_{2t_b - 2u + 1, 3t_b - 2u}^{m_2}. \quad (13.3)$$

Though we focussed on specific set of  $t_b$  shares, namely  $\{(2t_b - 2u + 1)^{\text{th}}, \dots, (3t_b - 2u)^{\text{th}}\}$ , the above relation should hold for any selection of  $t_b$  shares. Also, from the property of a  $(n, (n - 2(t_b - u)), t_b, m, \ell)$ -SSS, any set of  $n - 2(t_b - u)$  shares have *full* information about  $m$  and uniquely determine  $m$ . Thus,

$$\mathbf{M}_{2t_b - 2u + 1, n}^{m_1} \cap \mathbf{M}_{2t_b - 2u + 1, n}^{m_2} = \emptyset. \quad (13.4)$$

Again, though we focussed on specific set of  $n - (2t_b - 2u)$  shares, namely  $\{(2t_b - 2u + 1)^{\text{th}}, \dots, n^{\text{th}}\}$ , the above relation should hold for any selection of  $n - (2t_b - 2u)$  shares. From (13.3),  $\mathbf{M}_{2t_b - 2u + 1, 3t_b - 2u}^m$  will be same for all  $m$ . Thus, (13.4) will hold only if  $\mathbf{M}_{3t_b - 2u + 1, n}^m$  is *unique* for every  $m$ . Hence,

$$|\mathbf{M}_{3t_b - 2u + 1, n}| = |\mathcal{M}|. \quad (13.5)$$

From the definition of  $\mathbf{X}_i$  and  $\mathbf{M}_{i,j}$ , we get  $\prod_{i=3t_b - 2u + 1}^n |\mathbf{X}_i| \geq |\mathbf{M}_{3t_b - 2u + 1, n}|$ . Combining this with (13.5), we get

$$\prod_{i=3t_b - 2u + 1}^n |\mathbf{X}_i| \geq |\mathcal{M}|. \quad (13.6)$$

Let  $g = n - (3t_b - 2u)$ . The inequality in (13.6) holds for any set of  $g$  shares  $\mathcal{D}$ , where  $|\mathcal{D}| = g$ ; i.e.,  $\prod_{i \in \mathcal{D}} |\mathbf{X}_i| \geq |\mathcal{M}|$ . In particular, we consider  $n$  such sets (consisting of  $g$  shares), namely  $\mathcal{D}_0, \dots, \mathcal{D}_{n-1}$  where  $\mathcal{D}_k$  consists of  $\{(kg + 1)^{\text{th}} \bmod n, (kg + 2)^{\text{th}} \bmod n, \dots, (kg + g)^{\text{th}} \bmod n\}$  shares. Thus for each  $\mathcal{D}_k$ ,  $\prod_{i \in \mathcal{D}_k} |\mathbf{X}_i| \geq |\mathcal{M}|$  holds. Taking product over all  $\mathcal{D}_k$ 's, we obtain  $\prod_{k=0}^{n-1} \prod_{j \in \mathcal{D}_k} |\mathbf{X}_j| \geq |\mathcal{M}|^n$ . Now notice that the  $i^{\text{th}}$  share is accounted exactly  $g$  times in total in  $\mathcal{D}_0, \dots, \mathcal{D}_{n-1}$ . Thus, we get  $|\mathcal{M}|^n \leq \prod_{k=0}^{n-1} \prod_{j \in \mathcal{D}_k} |\mathbf{X}_j| = (\prod_{i=1}^n |\mathbf{X}_i|)^g$ . Taking log, we obtain

$$\begin{aligned} n \log(|\mathcal{M}|) &\leq g \sum_{i=1}^n \log(|\mathbf{X}_i|) \\ \Rightarrow \sum_{i=1}^n \log(|\mathbf{X}_i|) &\geq \left( \frac{n\ell \log(|\mathbb{F}|)}{g} \right) \end{aligned}$$

As  $\log(|\mathcal{M}|) = \ell \log(|\mathbb{F}|)$  and  $g = n - (3t_b - 2u)$ , from the above inequality, we get  $\sum_{i=1}^n \log(|\mathbf{X}_i|) \geq \left( \frac{n\ell \log(|\mathbb{F}|)}{n - (3t_b - 2u)} \right)$ . As mentioned earlier,  $\sum_{i=1}^n \log(|\mathbf{X}_i|)$  denotes the share complexity in bits of distributing  $n$  shares of a message  $m$  using any  $(n, (n - 2(t_b - u)), t_b, m, \ell)$ -SSS. From the above inequality, the share complexity of  $(n, (n - 2(t_b - u)), t_b, m, \ell)$ -SSS is  $\Omega\left(\frac{n\ell \log(|\mathbb{F}|)}{n - (3t_b - 2u)}\right)$  bits. Now each field element from  $\mathbb{F}$  can be represented by  $\log(|\mathbb{F}|)$  bits. Thus the share complexity is  $\Omega\left(\frac{n\ell}{n - (3t_b - 2u)}\right)$  field elements.

Part 1 of Theorem 13.20 now follows from Lemma 13.21 and Lemma 13.22. Now part 2 simply follows from the fact that any PSMT protocol has to send at least the message and hence  $\Omega(\ell)$  field elements.  $\square$

In the light of Theorem 13.20, we now state the following theorems:

**Theorem 13.23** *Protocol 3-Optimal-PSMT-Static-Byzantine-Directed is a communication optimal PSMT protocol.*

PROOF: From Theorem 13.12, if  $n = 3t_b - 2u + 1$  and  $0 < u \leq \frac{t_b}{2}$ , then protocol 3-Optimal-PSMT-Static-Byzantine-Directed sends a message  $m^{\mathbf{S}}$  containing  $\ell = n^2u$  field elements by communicating  $\mathcal{O}(n^3u) = \mathcal{O}\left(\frac{n\ell}{n-(3t_b-2u)}\right) = \mathcal{O}(n\ell)$  field elements. From Theorem 13.20, if  $n = 3t_b - 2u + 1$  and  $0 < u \leq \frac{t_b}{2}$ , then any three phase PSMT protocol must communicate  $\Omega\left(\frac{n\ell}{n-(3t_b-2u)}\right) = \Omega(n\ell)$  field elements to securely send a message containing  $\ell = n^2u$  field elements. Thus protocol 3-Optimal-PSMT-Static-Byzantine-Directed is a communication optimal PSMT protocol.  $\square$

**Theorem 13.24** *Protocol 6-Optimal-PSMT-Static-Byzantine-Directed is a communication optimal PSMT protocol.*

PROOF: From Theorem 13.15, if  $n = 2t_b + 1$  and  $\frac{t_b}{2} < u \leq t_b$ , then protocol 6-Optimal-PSMT-Static-Byzantine-Directed sends a message  $m^{\mathbf{S}}$  containing  $\ell = n^2u$  field elements by communicating  $\mathcal{O}\left(\frac{n^3u}{2u-t_b}\right) = \mathcal{O}\left(\frac{n\ell}{n-(3t_b-2u)}\right) = \mathcal{O}\left(\frac{n\ell}{2u-t_b+1}\right)$  field elements. From Theorem 13.20, if  $n = 2t_b + 1$  and  $\frac{t_b}{2} < u \leq t_b$ , then any six phase PSMT protocol must communicate  $\Omega\left(\frac{n\ell}{n-(3t_b-2u)}\right) = \Omega\left(\frac{n\ell}{2u-t_b+1}\right)$  field elements to securely send a message containing  $\ell = n^2u$  field elements. Thus protocol 6-Optimal-PSMT-Static-Byzantine-Directed is a communication optimal PSMT protocol for the case when  $\frac{t_b}{2} < u \leq t_b$ .

On the other hand, if  $n = 2t_b + 1$  and  $u > t_b$  then from Theorem 13.16, protocol 6-Optimal-PSMT-Static-Byzantine-Directed securely sends a message containing  $\Theta(n^3)$  field elements by communicating  $\mathcal{O}(n^3)$  field elements. From Theorem 13.20, if  $n = 2t_b + 1$  and  $u > t_b$  then any six phase PSMT protocol must communicate  $\Omega(n^3)$  field elements to securely send a message containing  $n^3$  field elements. Thus protocol 6-Optimal-PSMT-Static-Byzantine-Directed is a communication optimal PSMT protocol for the case when  $u > t_b$ .  $\square$

## 13.9 Concluding Remarks and Open Problems

In this chapter, we re-visited the PSMT problem in directed synchronous network, tolerating  $\mathcal{A}_{t_b}^{static}$ . Specifically, we presented new PSMT protocols in directed graphs tolerating  $\mathcal{A}_{t_b}^{static}$ , which are secure against guessing attack of [88]. Moreover, we also showed that the communication complexity of our protocols are asymptotically optimal. This completely settles the issue of POSSIBILITY, FEASIBILITY and OPTIMALITY of PSMT protocols in directed synchronous network, tolerating  $\mathcal{A}_{t_b}^{static}$ . The summary of our results (marked with \*) is as follows:

# Phases	Characterization	Lower Bound on Communication Complexity
1	$n \geq 3t_b + 1$ [28]	$\Omega\left(\frac{n\ell}{n-3t_b}\right)$ [30, 81]
2	If $0 < u \leq t_b$ then $n \geq 3t_b - u + 1^*$ If $u > t_b$ then $n \geq 2t_b + 1^*$	$\Omega\left(\frac{N\ell}{N-3t_b}\right)$ ; $N = n + u^*$ $\Omega\left(\frac{n\ell}{n-2t_b}\right)^*$
3	If $0 < u \leq t_b$ then $n \geq \max(3t_b - 2u + 1, 2t_b + 1)$ [24] If $u > t_b$ then $n \geq 2t_b + 1$ [24]	$\Omega\left(\frac{n\ell}{n-(3t_b-2u)}\right)^*$ $\Omega(\ell)^*$

**Open Problem 14** *This paper leaves few open problems which are as follows:*

1. *Protocol 2-Optimal-PSMT-Static-Byzantine-Directed, 3-Optimal-PSMT-Static-Byzantine-Directed and 6-Optimal-PSMT-Static-Byzantine-Directed are communication optimal only for sufficiently large messages. It would be interesting to design communication optimal PSMT protocols, which are communication optimal for messages of any length.*
2. *Reducing the phase complexity of protocol 6-Optimal-PSMT-Static-Byzantine-Directed is another interesting problem.*
3. *In this paper, we have only considered static and Byzantine adversary. It would be interesting to consider the problem of PSMT in directed synchronous network in other adversarial settings such as mobile adversary and mixed adversary.*

Figure 13.3: Three Phase PSMT Protocol:  $\ell = n^2u, n = 3t_b - 2u + 1, 0 < u \leq \frac{t_b}{2}$ ;  $\mathcal{C}$  is the set of all possible RS codewords of length  $n$  encoded by polynomials of degree  $t_b$

<b>Protocol 3-Optimal-PSMT-Static-Byzantine-Directed</b> ( $m^S, \ell, n, u, t_b$ )
<p><b>Phase I: S to R:</b></p> <ol style="list-style-type: none"> <li>1. <b>S</b> selects <math>P = n^2u + ut_b = \ell + ut_b</math> random codewords <math>C_1, \dots, C_P</math> from <math>\mathcal{C}</math>.</li> <li>2. For <math>i = 1, \dots, P</math>, let <math>C_i = (c_{i1}, \dots, c_{in})</math> and <math>F_i(x)</math> be the corresponding <math>t_b</math> degree polynomial.</li> <li>3. For <math>i = 1, \dots, P</math>, <b>S</b> sends <math>c_{ij}</math> to <b>R</b> along wire <math>f_j</math>, for <math>j = 1, \dots, n</math>.</li> </ol> <p><b>Phase II: R to S</b></p> <ol style="list-style-type: none"> <li>1. For <math>i = 1, \dots, P</math>, let <b>R</b> receive the vector <math>Y_i = C_i + E_i</math> corresponding to codeword <math>C_i</math> and let <math>\mathcal{Y} = \{Y_1, \dots, Y_P\}</math>. Here <math>E_i</math> is the error introduced in <math>C_i</math>.</li> <li>2. <b>R</b> invokes <math>(p, \mathcal{B}, \mathcal{I}) = \text{Find-Pseudo-basis}(\mathcal{Y})</math> to find pseudo-basis <math>\mathcal{B} = \{Y_{a_1}, \dots, Y_{a_p}\} \subset \mathcal{Y}</math>, pseudo-dimension <math>p =  \mathcal{B} </math> and index set <math>\mathcal{I} = \{a_1, \dots, a_p\} \subset \{1, \dots, P\}</math>.</li> <li>3. <b>R</b> broadcasts <math>(\mathcal{B}, p, \mathcal{I})</math> to <b>S</b> through the <i>bottom band</i>.</li> </ol> <p><b>Phase III: S to R</b></p> <ol style="list-style-type: none"> <li>1. <b>S</b> may receive different triples over different wires. For <math>j = 1, \dots, u</math>, let <b>S</b> receive <math>(\mathcal{B}^j, p^j, \mathcal{I}^j)</math> over wire <math>b_j</math> in <i>bottom band</i>. Let <math>\mathcal{B}^j = \{Y_{a_1^j}, \dots, Y_{a_{p^j}^j}\}</math> and <math>\mathcal{I}^j = \{a_1^j, \dots, a_{p^j}^j\}</math>.</li> <li>2. <b>S</b> considers the triple <math>(\mathcal{B}^j, p^j, \mathcal{I}^j)</math> as <b>valid</b> iff <math>p^j =  \mathcal{B}^j </math>, <math>p^j \leq t_b</math> and every vector of length <math>n</math> listed in <math>\mathcal{B}^j</math> is different from the corresponding original codeword in at most <math>t_b</math> locations.</li> <li>3. For every <b>valid</b> triple <math>(\mathcal{B}^j, p^j, \mathcal{I}^j)</math>, <b>S</b> finds <math>E_{a_1^j}^j = Y_{a_1^j}^j - C_{a_1^j}</math>, <math>\dots</math>, <math>E_{a_{p^j}^j}^j = Y_{a_{p^j}^j}^j - C_{a_{p^j}^j}</math> and computes <math>\text{CORRUPTED}^j = \cup_{\alpha=1}^{p^j} \text{support}(E_{a_\alpha^j}^j)</math>.</li> <li>4. <b>S</b> computes <math>\Lambda = \cup_j \{\mathcal{I}^j   (\mathcal{B}^j, p^j, \mathcal{I}^j) \text{ is a valid triple}\}</math>. Then <b>S</b> concatenates all the <math>F_i(0)</math>'s such <math>i \notin \Lambda</math> and forms an information theoretic secure pad <math>Z</math> of length at least <math>n^2u</math> (since <math> \Lambda  \leq ut_b</math> and <math>P = n^2u + ut_b</math>, this is possible).</li> <li>5. Now <b>S</b> broadcasts the following to <b>R</b>: <ol style="list-style-type: none"> <li>(a) If there is no <b>valid</b> triple, then the message “<b>Entire Bottom band is corrupted</b>”;</li> <li>(b) Every <b>valid</b> triple <math>(\mathcal{B}^j, p^j, \mathcal{I}^j)</math> and the corresponding list of corrupted wires <math>\text{CORRUPTED}^j</math>;</li> <li>(c) Blinded message <math>\Gamma = Z_\ell \oplus m^S</math> where <math>Z_\ell</math> contains first <math>\ell</math> elements from <math>Z</math>.</li> </ol> </li> </ol> <p><b>Local Computation by R at the End of Phase III:</b></p> <ol style="list-style-type: none"> <li>1. <b>R</b> correctly receives all information sent by <b>S</b> in <b>Phase III</b> and computes <math>\Lambda</math> in same way as done by <b>S</b>.</li> <li>2. If either <b>R</b> gets the message “<b>Entire Bottom band is corrupted</b>” or if <b>R</b> finds his original triple <math>(\mathcal{B}, p, \mathcal{I})</math> is not present in the list of <b>valid</b> triples sent by <b>S</b>, then <b>R</b> does the following: <ol style="list-style-type: none"> <li>(a) Conclude that entire <i>bottom band</i> is corrupted and hence in the top band there are at most <math>t_b - u</math> faults.</li> <li>(b) Recover all <math>F_i(x)</math> such that <math>i \notin \Lambda</math> by executing RS-DEC(<math>n, Y_i, t_b - u, 0, t_b + 1</math>).</li> <li>(c) Recover pad <math>Z</math> (and hence <math>Z_\ell</math>) by concatenating <math>F_i(0)</math>'s for all <math>i \notin \Lambda</math>.</li> <li>(d) Recover <math>m^S = \Gamma \oplus Z_\ell</math>, where <math>\Gamma</math> is received correctly from <b>S</b>.</li> </ol> </li> <li>3. If <b>R</b> finds that his original triple <math>(\mathcal{B}, p, \mathcal{I})</math> is present in the list of <b>valid</b> triples sent by <b>S</b>, then <b>R</b> does following: <ol style="list-style-type: none"> <li>(a) Let <math>(\mathcal{B}^j, p^j, \mathcal{I}^j)</math> be same as <math>(\mathcal{B}, p, \mathcal{I})</math>.</li> <li>(b) Identify all the wires in <math>\text{CORRUPTED}^j</math> (<math> \text{CORRUPTED}^j  \leq t_b</math>) as the corrupted wires in <b>Phase I</b>.</li> <li>(c) Ignore all information received over the wires in <math>\text{CORRUPTED}^j</math> (<math> \text{CORRUPTED}^j  \leq t_b</math>) during <b>Phase I</b>.</li> <li>(d) Reconstruct all the polynomial <math>F_i(x)</math> such that <math>i \notin \Lambda</math> by considering the correct values of <math>F_i(x)</math> received over the remaining wires (which are at least <math>t_b + 1</math>) during <b>Phase I</b>.</li> <li>(e) Recover <math>m^S</math> in the same way as described in step 2 of <b>Local Computation</b>.</li> </ol> </li> </ol>

Figure 13.4: Six Phase PSMT Protocol:  $n = 2t_b + 1$ ,  $\frac{t_b}{2} < u \leq t_b$ ,  $\ell = n^2 u$ ;  $\mathcal{C}$  denotes the set of all possible RS codewords of length  $N = n + u = 2t_b + 1 + u$  encoded using all possible polynomials of degree  $2u > t_b$

**Protocol 6-Optimal-PSMT-Static-Byzantine-Directed( $m^S, \ell, n, u, t_b$ )**

**Phase I: R to S:** **R** selects  $Q = \frac{n^2 u}{2u - t_b + 1} + ut_b = \frac{\ell}{2u - t_b + 1} + ut_b$  random vectors  $R_1, \dots, R_Q$  of length  $u$ , such that  $R_i = (r_{i1}, \dots, r_{iu})$ . Now **R** sends the  $j^{\text{th}}$  component of all the vectors to **S** along wire  $b_j$ .

**Phase II: S to R:** **S** receives  $\bar{R}_1, \dots, \bar{R}_Q$  and selects  $Q$  codewords  $C_1, \dots, C_Q$  from  $\mathcal{C}$  such that the last  $u$  components of  $C_i$  are same as in  $\bar{R}_i$ . This is always possible because every codeword  $C_i$  corresponds to a  $2u$  degree polynomial  $F_i(x)$ . Now **S** sends the  $j^{\text{th}}$  component of all the codewords to **R** over wire  $f_j$ .

**Phase III: R to S**

1. After receiving information over the top band, **R** possesses a vector of length  $N$  (by combining the values sent over the bottom band and the values received over the top band)  $Y_i = C_i + E_i$  corresponding to  $C_i$ , such that  $Y_i$  is different from  $C_i$  in at most  $t_b$  locations. Let  $\mathcal{Y} = \{Y_1, \dots, Y_Q\}$ .
2. Now **R** does same computation and communication as in **Phase II** of protocol 3-Optimal-PSMT-Static-Byzantine-Directed. The only difference is that here  $\mathcal{Y}$  contains vectors  $\{Y_1, \dots, Y_Q\}$  of length  $N = 2t_b + 1 + u$ , whereas in 3-Optimal-PSMT-Static-Byzantine-Directed,  $\mathcal{Y}$  contains vectors  $\{Y_1, \dots, Y_P\}$  of length  $n = 3t - 2u + 1$ . Notice that FindPseudo-basis will still be able to find out pseudo-basis. This is because the code  $\mathcal{C}$  used here has a hamming distance of at least  $t_b + 1$ .

**Phase IV: S to R**

1. With respect to the triples received through the bottom band, **S** performs the same computation (not communication) as done in **Phase III** of Protocol 3-Optimal-PSMT-Static-Byzantine-Directed. That means **S** identifies the valid triples and for each valid triple  $(\mathcal{B}^j, p^j, \mathcal{I}^j)$ , **S** finds the list of corrupted wires  $CORRUPTED^j$ . But here are following differences: (i) the pad  $Z$  is generated in a different manner, (ii) the valid triples, their corresponding list of corrupted wires and the blinded message are sent in a different manner.
2. **Generation of pad  $Z$ :**
  - (a) **S** computes  $\Lambda = \cup_j \{\mathcal{I}^j | (\mathcal{B}^j, p^j, \mathcal{I}^j) \text{ is a valid triple}\}$ .
  - (b) **S** computes  $Z^i = (z_1^i, \dots, z_{2u - t_b + 1}^i) = \text{EXTRAND}_{N, 2u - t_b + 1}(C_i)$  for each  $i \notin \Lambda$ .
  - (c) Since  $|\Lambda| \leq ut_b$  and  $Q = \frac{n^2 u}{2u - t_b + 1} + ut_b$ , **S** has generated at least  $\frac{n^2 u}{2u - t_b + 1}$   $Z^i$ 's. Hence concatenating all  $Z^i$ , **S** obtains a pad  $Z$  of length at least  $n^2 u$ .
3. **Communication done by S:**
  - (a) If there is no **valid** triple, then **S** simply broadcasts the message “**Entire Bottom band is corrupted**” over all the wires in *top band*.
  - (b) **S** merges all the quadruples  $(\mathcal{B}^j, p^j, \mathcal{I}^j, CORRUPTED^j)$  such that  $(\mathcal{B}^j, p^j, \mathcal{I}^j)$  is a valid triple into a list called  $L$  and sends it to **R** reliably by executing the protocol 3-Optimal-PRMT-Static-Byzantine-Directed.
  - (c) **S** sends the blinded message  $\Gamma = Z_\ell \oplus m^S$  by executing another instance of protocol 3-Optimal-PRMT-Static-Byzantine-Directed where  $Z_\ell$  contains first  $\ell$  elements from  $Z$ .
  - (d) Since  $n = 2t_b + 1$  and  $n - 2t_b + 2u = \Theta(n)$ , 3-Optimal-PRMT-Static-Byzantine-Directed sends the message in three phases. **R** receives all information communicated by **S** during **Phase IV** at the end of **Phase VI**.

**Local Computation by R At The End of Phase VI:**

1. **R** correctly receives all the information that **S** had sent during **Phase IV** and computes  $\Lambda$  in the same manner as done by **S**.
2. If either **R** gets the message “**Entire Bottom band is corrupted**” or if **R** finds his original triple  $(\mathcal{B}, p, \mathcal{I})$  is not present in the list of **valid** triples sent by **S**, then **R** does the following:
  - (a) Conclude that entire *bottom band* is corrupted and hence in the *top band* there are at most  $t_b - u$  faults.
  - (b) Neglect last  $u$  components of all  $Y_i$  and then recover all  $C_i$  such that  $i \notin \Lambda$  by applying RS decoding algorithm on truncated  $Y_i$  and correcting  $t_b - u$  Byzantine faults.
  - (c) Compute pad  $Z$  in the same way as done by **S** and recovers  $m^S = \Gamma \oplus Z_\ell$ .
3. If **R** finds that his original triple  $(\mathcal{B}, p, \mathcal{I})$  is present in the list of **valid** triples sent by **S** and if  $(\mathcal{B}^j, p^j, \mathcal{I}^j)$  is same as  $(\mathcal{B}, p, \mathcal{I})$ , then **R** does the following:
  - (a) Identify all the wires in  $CORRUPTED^j$  as the corrupted wires.
  - (b) Ignore all information communicated over the wires in  $CORRUPTED^j$ . Reconstruct all  $C_i$  such that  $i \notin \Lambda$ . This is possible because  $|CORRUPTED^j| \leq t_b$ . Hence  $N - |CORRUPTED^j| \geq (t_b + 1 + u) \geq 2u + 1$  and each codeword  $C_i$  is encoded using a polynomial of degree  $2u$ .
  - (c) Recover the message  $m^S$  in the same way as described in step 2 of **Local Computation**.

## Chapter 14

# SSMT in Undirected Networks Tolerating Static Mixed Adversary

In this chapter, we study SSMT in *undirected synchronous network* tolerating *threshold static mixed adversary*  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . The characterization for single phase and multi phase SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is given in [75]. In [75], the author has also derived the lower bound on the communication complexity of single and multi phase SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . Moreover, the author also presented a single phase communication optimal SSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , thus completely resolving the issue of POSSIBILITY and OPTIMALITY of single phase SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . However, no multi phase communication optimal SSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  was presented. This left the problem of designing multi phase communication optimal SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  as open. In this chapter, we settle this problem by designing a communication optimal SSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . We now briefly describe the network model and adversary settings used in this chapter.

### 14.1 Network Model and Adversary Settings

The network model used in this chapter is similar to the one used in Chapter 11. Thus, we assume that  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n$  parallel and synchronous bi-directional node disjoint paths/channels  $w_1, w_2, \dots, w_n$ , also called as *wires*. The adversary  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , having unbounded computing power can corrupt up to  $t_b, t_f$  and  $t_p$  wires in Byzantine, failstop and passive fashion respectively. Moreover, as a worst case assumption, we assume that the wires that are under the control of the adversary in Byzantine, failstop and passive fashion are mutually disjoint.

All our SSMT protocols will have a negligible error probability of  $2^{-\Omega(\kappa)}$  in reliability. To bound the error probability by  $2^{-\Omega(\kappa)}$ , all computation and communication in our protocols are performed over a finite field  $\mathbb{F} = GF(2^\kappa)$ . Thus each field element from  $\mathbb{F}$  can be represented by  $\mathcal{O}(\kappa)$  bits. Moreover, without loss of generality, we assume that  $n = \text{poly}(\kappa)$ .

We now recall the existing results for SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  from [75].

## 14.2 Existing Results for SSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

We now recall the existing results for SSMT in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . We first begin with single phase SSMT.

### 14.2.1 Existing Results for Single Phase SSMT

SSMT problem was first introduced in [33], where the authors studied SSMT in undirected synchronous network tolerating threshold static Byzantine adversary  $\mathcal{A}_{t_b}^{static}$ . The authors in [33] gave the following characterization.

**Theorem 14.1 ([33])** *Let  $r \geq 1$ . Then any  $r$ -phase SSMT protocol in undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff there exists  $n \geq 2t_b + 1$  wires between **S** and **R**.*

Extending the above result for the case of mixed adversary  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , Srinathan [75] gave the following characterization:

**Theorem 14.2 ([75])** *Any single phase SSMT protocol in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is possible iff there exists  $n \geq 2t_b + t_f + t_p + 1$  wires between **S** and **R**.*

The lower bound on the communication complexity of single phase SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is given by the following theorem.

**Theorem 14.3 ([75])** *Let **S** and **R** be connected by  $n \geq 2t_b + t_f + t_p + 1$  wires. Then any single phase SSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , must communicate  $\Omega\left(\frac{n\ell}{n-(2t_b+t_f+t_p)}\right)$  field elements to securely send a message containing  $\ell$  field elements. In terms of bits, any single phase SSMT must communicate  $\Omega\left(\frac{n\ell\kappa}{n-(2t_b+t_f+t_p)}\right)$  bits to securely send a message containing  $\ell\kappa$  bits.*

PROOF (SKETCH): The lower bound is derived by first arguing that the communication done in any single phase SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  to send a message  $m$  containing  $\ell$  field elements, is not less than the share complexity (sum of the length of the shares) of generating  $n$  shares for  $m$ , such that any set of  $(t_b + t_p)$  shares has no information about  $m$ , while any set of  $n - (t_b + t_f)$  shares has full information about  $m$ . Then it is shown that share complexity of any secret sharing scheme with the above mentioned property is at least  $\Omega\left(\frac{n\ell}{n-(2t_b+t_f+t_p)}\right)$  field elements.  $\square$

**Comparison 14.4 (Possibility of Single Phase PSMT and SSMT)** *From Theorem 11.1, single phase PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is possible iff there exists  $n \geq 3t_b + t_f + t_p + 1$  wires between **S** and **R**. But from Theorem 14.2, we find that single phase SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is possible iff there exists  $n \geq 2t_b + t_f + t_p + 1$  wires between **S** and **R**. This shows that allowing a negligible error probability (only in the reliability), significantly helps in the POSSIBILITY of single phase secure message transmission protocols.*

**Comparison 14.5 (Complexity of Single Phase SSMT and PSMT)** *From Theorem 11.3, any single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  over  $n \geq 3t_b + t_f + t_p + 1$  wires has to communicate  $\Omega\left(\frac{n\ell}{n-(3t_b+t_f+t_p)}\right)$  field elements to securely send a message containing*

$\ell$  field elements. From Theorem 14.3, any single phase SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  over  $n \geq 2t_b + t_f + t_p + 1$  wires has to communicate  $\Omega\left(\frac{n\ell}{n-(2t_b+t_f+t_p)}\right)$  field elements to securely send a message containing  $\ell$  field elements. Let us fix  $n = 3t_b + t_f + t_p + 1$  such that both PSMT and SSMT is possible (notice that with  $n = 2t_b + t_f + t_p + 1$  wires, SSMT is possible but PSMT is not possible). With  $n = 3t_b + t_f + t_p + 1$ , the lower bounds for PSMT and SSMT become  $\Omega(n\ell)$  and  $\Omega\left(\frac{n\ell}{t_b}\right)$  field elements respectively. Specifically, if we consider  $\mathcal{A}_{t_b}^{static}$  then  $n$  must be at least  $3t_b + 1$  for PSMT to be possible (notice that SSMT requires only  $2t_b + 1$  wires for tolerating  $\mathcal{A}_{t_b}^{static}$ ). With  $n = 3t_b + 1$ , the lower bounds for single phase PSMT and SSMT become  $\Omega(n\ell)$  and  $\Omega(\ell)$  field elements respectively, for now  $t_b = \Theta(n)$ . Hence with  $n = 3t_b + 1$  while SSMT can be achieved with constant factor overhead tolerating  $\mathcal{A}_{t_b}^{static}$ , PSMT can not be achieved. This shows the power of allowing a negligible error probability (only in the reliability) in single phase secure message transmission.

In [75], the author presented a single phase SSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , which asymptotically satisfies the bound given in Theorem 14.3. For the sake of completeness, we recall this protocol. Another reason for recalling the protocol is that it will be used as a black box in our multi phase communication optimal SSMT protocol (which is the main contribution of this chapter).

#### 14.2.1.1 Single Phase Communication Optimal SSMT Against $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ [75]

In [75], the author presented a single phase communication optimal SSMT protocol in the presence of  $n = 2t_b + t_f + t_p + 1$  wires, which we call 1-Optimal-SSMT-Static-Mixed. The protocol tolerates  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  and securely sends a message containing  $(t_b + t_f + t_p)\kappa = \Theta(n\kappa)$  bits by communicating  $\mathcal{O}(n^2\kappa)$  bits. Let  $m^S$  be the secret message containing  $t_b + t_f + t_p$  field elements. The protocol uses Extrapolation Technique (see Fig. 8.1) and is formally presented in Fig. 14.1.

We now prove the properties of protocol 1-Optimal-SSMT-Static-Mixed.

**Lemma 14.6** *In protocol 1-Optimal-SSMT-Static-Mixed if  $\mathbf{R}$  concludes that  $F_i^{\mathbf{R}}$  is a valid row of  $B^{\mathbf{S}}$ , then except with error probability  $2^{-\Omega(\kappa)}$ ,  $F_i^{\mathbf{R}} = F_i^{\mathbf{S}}$ .*

PROOF: The lemma is true without any error if wire  $w_i$  is uncorrupted. So let wire  $w_i$  be a corrupted wire, who delivers  $F_i^{\mathbf{R}} \neq F_i^{\mathbf{S}}$ . In this case, if  $F_i^{\mathbf{R}}$  is considered as a valid row of  $B^{\mathbf{S}}$ , then it implies that  $Support_i \geq t_b + t_p + 1$ . Now out of these  $(t_b + t_p + 1)$  wires, at most  $t_b + t_p$  wires can be under the control of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  in Byzantine and passive fashion respectively. This implies that there exists at least one wire, say  $w_j$ , which correctly and securely delivered the hash key  $\alpha_j^{\mathbf{R}} = \alpha_j^{\mathbf{S}}$  and hash value  $v_{ij}^{\mathbf{R}} = v_{ij}^{\mathbf{S}} = hash(\alpha_j^{\mathbf{S}}; F_i^{\mathbf{S}}) = hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{S}})$ , such that  $w_j \in Support_i$ . Since  $w_j \in Support_i$ , it implies that  $v_{ij}^{\mathbf{R}} = hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{R}})$ . Since adversary does not know  $\alpha_j^{\mathbf{R}}$  and  $v_{ij}^{\mathbf{R}}$ , he can ensure that  $v_{ij}^{\mathbf{R}} = hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{S}})$ , as well as  $v_{ij}^{\mathbf{R}} = hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{R}})$ , where  $F_i^{\mathbf{R}} \neq F_i^{\mathbf{S}}$ , with probability at most  $\frac{n+t_p-1}{|\mathbb{F}|} \approx 2^{-\Omega(\kappa)}$ , which is negligible in our context. So with very high probability,  $w_j$  will not belong to  $Support_i$ , which is a contradiction. So with overwhelming probability  $F_i^{\mathbf{R}} = F_i^{\mathbf{S}}$ .  $\square$

**Lemma 14.7** *In protocol 1-Optimal-SSMT-Static-Mixed, if  $\mathbf{R}$  gets  $t_b + t_p + 1$  valid rows of  $B^{\mathbf{S}}$  then  $\mathbf{R}$  can recover  $m^{\mathbf{S}}$ .*

Figure 14.1: Single Phase Communication Optimal SSMT Protocol Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ :  $n = 2t_b + t_f + t_p + 1, |m^{\mathbf{S}}| = (t_b + t_f + t_p)$

**Protocol 1-Optimal-SSMT-Static-Mixed**

**Computation and Communication by S:**

1. **S** sets  $ROW = t_b + t_p + 1, COL = n + t_p$  and  $N = n$ . **S** then forms a matrix  $B_{init}$ , consisting of  $ROW \times COL$  random, non-zero elements from  $\mathbb{F}$ . **S** then executes the steps in **Extrapolation Technique** to generate an  $N \times COL = n \times (n + t_p)$  matrix  $B_{ext}$  from  $B_{init}$ . Let  $B^{\mathbf{S}} = B_{ext}$ .
2. For  $i = 1, \dots, n$ , **S** selects a random, secret, non-zero hash key  $\alpha_i^{\mathbf{S}}$ , corresponding to wire  $w_i$ .
3. For  $i = 1, \dots, n$ , **S** sends the following to **R** over wire  $w_i$ :
  - (a) The  $i^{th}$  row of  $B^{\mathbf{S}}$ , denoted by  $F_i^{\mathbf{S}}$ ;
  - (b) The hash key  $\alpha_i^{\mathbf{S}}$  and
  - (c) The hash values  $v_{ji}^{\mathbf{S}}$ , where  $v_{ji}^{\mathbf{S}} = hash(\alpha_i^{\mathbf{S}}; F_j^{\mathbf{S}})$ , for  $j = 1, \dots, n$ .
4. **S** computes
$$\mathcal{P}^{\mathbf{S}} = \text{EXTRAND}_{n(n+t_p), t_b+t_f+t_p}(B^{\mathbf{S}}).$$
5. **S** then computes  $C = \mathcal{P}^{\mathbf{S}} \oplus m^{\mathbf{S}}$ , broadcasts  $C$  to **R** and terminates the protocol.

**Message Recovery by R:**

1. Let  $\mathcal{F}$  denote the set of wires that delivered nothing to **R**, where  $|\mathcal{F}| \leq t_f$ .
2. Let **R** receive the following over wire  $w_i \notin \mathcal{F}$ :
  - (a) The  $(n + t_p)$ -tuple, denoted by  $F_i^{\mathbf{R}}$ ;
  - (b) The hash key  $\alpha_i^{\mathbf{R}}$  and
  - (c) The hash values  $v_{ji}^{\mathbf{R}}$ , for  $j = 1, \dots, n$ .
3. For every  $w_i \notin \mathcal{F}$ , **R** computes  $Support_i = |\{w_j : w_j \notin \mathcal{F} \text{ and } v_{ij}^{\mathbf{R}} = hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{R}})\}|$ .
4. If  $Support_i \geq t_b + t_p + 1$ , then **R** considers  $F_i^{\mathbf{R}}$  to be a valid row of  $B^{\mathbf{S}}$ .
5. If **R** has received  $ROW = t_b + t_p + 1$  valid rows, then **R** reconstructs the message  $m^{\mathbf{R}}$  from them and terminates the protocol.

PROOF: The proof follows from Lemma 8.10 and the fact that  $ROW = t_b + t_p + 1$  in protocol 1-Optimal-SSMT-Static-Mixed.  $\square$

**Lemma 14.8** *In protocol 1-Optimal-SSMT-Static-Mixed, except with error probability  $2^{-\Omega(\kappa)}$ ,  $m^{\mathbf{R}} = m^{\mathbf{S}}$ .*

PROOF: First of all notice that  $\mathbf{R}$  will always output some message. This is because at most  $t_b$  wires may deliver invalid row and at most  $t_f$  wires may not deliver any row of  $B^{\mathbf{S}}$ . However there always exist  $n - (t_b + t_f) = t_b + t_p + 1$  wires, which will always correctly deliver valid rows of  $B^{\mathbf{S}}$ . Moreover, even if a wire which is under the control of the adversary in Byzantine fashion has delivered a valid row, then from Lemma 14.6, the row is indeed a valid row of  $B^{\mathbf{S}}$ , except with probability  $2^{-\Omega(\kappa)}$ . This implies that with very high probability, the  $t_b + t_p + 1$  valid rows used by  $\mathbf{R}$  to recover  $m^{\mathbf{R}}$  are indeed the rows of  $B^{\mathbf{S}}$ . Thus, except with error probability  $2^{-\Omega(\kappa)}$ ,  $m^{\mathbf{R}} = m^{\mathbf{S}}$ .  $\square$

**Lemma 14.9** *In protocol 1-Optimal-SSMT-Static-Mixed,  $m^{\mathbf{S}}$  will be information theoretically secure.*

PROOF: Notice that  $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{static}}$  can eavesdrop at most  $t_b + t_p$  wires. Without loss of generality, let these be the first  $t_b + t_p$  wires. So  $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{static}}$  will know the vectors  $F_1^{\mathbf{S}}, \dots, F_{t_b+t_p}^{\mathbf{S}}$ , from which it will come to know  $t_b + t_p$  rows of  $B^{\mathbf{S}}(x)$ , which are insufficient to reconstruct  $B^{\mathbf{S}}$ .

The adversary  $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{static}}$  will also know  $t_b + t_p$  hash values corresponding to each  $F_1^{\mathbf{S}}, \dots, F_n^{\mathbf{S}}$ . Since the vectors  $F_1^{\mathbf{S}}, \dots, F_{t_b+t_p}^{\mathbf{S}}$  are already known to  $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{static}}$ , the  $t_b + t_p$  hash values corresponding to them does not add anything new to  $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{static}}$ 's view. Moreover, since  $ROW = t_b + t_p + 1$ , from the properties of Extrapolation Technique (see Lemma 8.10) the vectors  $F_{t_b+t_p+2}^{\mathbf{S}}, \dots, F_n^{\mathbf{S}}$  can be expressed as a linear combination of vectors  $F_1^{\mathbf{S}}, \dots, F_{t_b+t_p+1}^{\mathbf{S}}$ . So the  $t_b + t_p$  hash values corresponding to  $F_{t_b+t_p+2}^{\mathbf{S}}, \dots, F_n^{\mathbf{S}}$  can always be expressed as a linear combination of the  $t_b + t_p$  hash values corresponding to  $F_1^{\mathbf{S}}, \dots, F_{t_b+t_p+1}^{\mathbf{S}}$ , which are known to the adversary. So, out of the  $t_b + t_p$  hash values corresponding to each  $F_i^{\mathbf{S}}, 1 \leq i \leq n$ , which are known to  $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{static}}$ , only the  $t_b + t_p$  hash values corresponding to  $F_{t_b+t_p+1}^{\mathbf{S}}$  add to  $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{static}}$ 's view. But  $F_{t_b+t_p+1}^{\mathbf{S}}$  is of length  $n + t_p$ . So from the properties of hashing,  $(n + t_p) - (t_b + t_p) = t_b + t_f + t_p + 1$  elements of  $F_{t_b+t_p+1}^{\mathbf{S}}$  will be information theoretically secure. This implies that  $t_b + t_f + t_p + 1$  elements of  $B^{\mathbf{S}}$  will be information theoretically secure. So from the properties of algorithm EXTRAND, the elements in  $\mathcal{P}^{\mathbf{S}}$  are information theoretically secure. This further implies information theoretic security of  $m^{\mathbf{S}}$ .  $\square$

**Lemma 14.10** *In protocol 1-Optimal-SSMT-Static-Mixed,  $\mathbf{S}$  communicates  $\mathcal{O}(n^2\kappa)$  bits.*

PROOF: Over each wire,  $\mathbf{S}$  sends a row of  $B^{\mathbf{S}}$  consisting of  $n + t_p$  elements, a hash key and  $n$  hash values. So overall,  $\mathbf{S}$  sends  $\mathcal{O}(n^2)$  field elements to  $\mathbf{R}$ . Since each field element can be represented by  $\mathcal{O}(\kappa)$  bits,  $\mathbf{S}$  communicates  $\mathcal{O}(n^2\kappa)$  bits.

**Theorem 14.11** *Protocol 1-Optimal-SSMT-Static-Mixed is a single phase communication optimal SSMT protocol, which securely sends a message containing  $\Theta(n\kappa)$  bits by communicating  $\mathcal{O}(n^2\kappa)$  bits, tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{static}}$ .*

PROOF: The proof that 1-Optimal-SSMT-Static-Mixed is a single phase SSMT protocol follows from Lemma 14.6, Lemma 14.7, Lemma 14.8 and Lemma 14.9. If  $n = 2t_b + t_f + t_p + 1$ , then from Theorem 14.3, any single phase SSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{static}}$  must communicate  $\Omega(n^2\kappa)$  bits to securely send  $\ell = (t_b + t_f + t_p) = \Theta(n\kappa)$  bits. Since the communication complexity of protocol 1-Optimal-SSMT-Static-Mixed is  $\mathcal{O}(n^2\kappa)$  bits, protocol 1-Optimal-SSMT-Static-Mixed is a communication optimal SSMT protocol.  $\square$

### 14.2.1.2 Protocol 1-Optimal-SSMT-Static-Mixed and Lower Bound on Communication Complexity of Single Phase SSMT Tolerating $\mathcal{A}_{t_b}^{static}$ [43]

Notice that any single phase SSMT tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff  $n \geq 2t_b + 1$ . Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + 1$  wires. Then from [43], for any single phase SSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , the following must hold

$$|\mathcal{X}_i| \geq \frac{|\mathcal{S} - 1|}{\delta} + 1 \quad (14.1)$$

where  $\mathcal{S}$  denotes the set of possible secret messages from which  $\mathbf{S}$  intends to send one element to  $\mathbf{R}$ ,  $\mathcal{X}_i$  denotes the set of possible data sent through the  $i^{th}$  wire in the protocol and  $0 < \delta < \frac{1}{2}$  is the error probability of the protocol. In any single phase SSMT protocol, one element from  $\mathcal{X}_i$  is sent through the  $i^{th}$  wire. Now each element of  $\mathcal{X}_i$  can be represented by  $\log(|\mathcal{X}_i|)$  bits. Similarly, each message from  $\mathcal{S}$  can be represented by  $\log(|\mathcal{S}|)$  bits. Thus inequality (14.1) says that any single phase SSMT protocol must communicate  $\Omega(n \log(|\mathcal{X}_i|))$  bits to securely send  $\log(|\mathcal{S}|)$  bits with error probability of at most  $0 < \delta < \frac{1}{2}$ .

In [43], the authors have proposed a near optimum single phase SSMT protocol whose total communication complexity *approximately* matches the bound given in inequality (14.1). However, the computation done by  $\mathbf{R}$  in their protocol is exponential in  $n$ . We now show that if we execute protocol 1-Optimal-SSMT-Static-Mixed in the presence of only  $\mathcal{A}_{t_b}^{static}$  over  $n = 2t_b + 1$  wires, then it satisfies the lower bound given in inequality (14.1).

If we execute protocol 1-Optimal-SSMT-Static-Mixed in the presence of  $\mathcal{A}_{t_b}^{static}$  over  $n = 2t_b + 1$  wires (i.e.,  $t_f = t_p = 0$ ), then the protocol securely sends  $t_b + 1 = \Theta(n)$  field elements (if  $n = 2t_b + 1$ , then  $t_b = \Theta(n)$ ) by communicating  $\mathcal{O}(n^2)$  field elements. Recall that each field element can be represented by  $\mathcal{O}(\kappa)$  bits. So in the presence of  $\mathcal{A}_{t_b}^{static}$ , the protocol will securely send  $\Theta(n\kappa)$  bits by communicating  $\mathcal{O}(n^2\kappa)$  bits.

We now show that the communication complexity of protocol 1-Optimal-SSMT-Static-Mixed in the presence of  $\mathcal{A}_{t_b}^{static}$  over  $n = 2t_b + 1$  wires satisfies the bound given in inequality (14.1). In the protocol, message space is  $\mathbb{F}^{t_b+1}$ . So  $\mathcal{S} = \mathbb{F}^{t_b+1}$  and thus  $\log(|\mathcal{S}|) = (t_b + 1) \log(|\mathbb{F}|) = (t_b + 1)\kappa$ . Substituting  $\delta = 2^{-\kappa}$  and value of  $\mathcal{S}$  in inequality (14.1), we get  $|\mathcal{X}_i| \geq \frac{|\mathbb{F}^{t_b+1}| - 1}{2^{-\kappa}} + 1$  and thus  $\log(|\mathcal{X}_i|) \geq \kappa + (t_b + 1)\kappa$ . So according to the lower bound given by inequality (14.1), any single phase SSMT over  $n = 2t_b + 1$  wires in the presence of  $\mathcal{A}_{t_b}^{static}$  must communicate  $\Omega(n(t_b + 1)\kappa) = \Omega(n^2\kappa)$  bits to securely send  $(t_b + 1)\kappa = \Theta(n\kappa)$  bits. However, the total communication complexity of protocol 1-Optimal-SSMT-Static-Mixed in the presence of  $\mathcal{A}_{t_b}^{static}$  over  $n = 2t_b + 1$  wires is  $\mathcal{O}(n^2\kappa)$  bits. Thus the protocol will be an efficient, single phase communication optimal SSMT tolerating  $\mathcal{A}_{t_b}^{static}$ .

In the next section, we present the existing results for multi phase SSMT in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .

### 14.2.2 Existing Results for Multi Phase SSMT

In [75], the author gave the following characterization for multi phase SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .

**Theorem 14.12 ([75])** *Let  $r \geq 2$ . Then any  $r$ -phase SSMT protocol in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is possible iff there exists  $n \geq t_b + t_f + \max(t_p, t_b) + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ .*

The lower bound on the communication complexity of multi phase SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is given by the following theorem, which is taken from [75].

**Theorem 14.13 ([75])** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq t_b + t_f + \max(t_p, t_b) + 1$  wires and let  $r \geq 2$ . Then any  $r$ -phase SSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , must communicate  $\Omega\left(\frac{n\ell}{n-(t_b+t_f+t_p)}\right)$  field elements to securely send a message containing  $\ell$  field elements. In terms of bits, any multi phase SSMT must communicate  $\Omega\left(\frac{n\ell\kappa}{n-(t_b+t_f+t_p)}\right)$  bits to securely send a message containing  $\ell\kappa$  bits.*

**Comparison 14.14 (POSSIBILITY OF MULTI PHASE PSMT AND SSMT)** *From Theorem 11.4, any  $r$ -phase PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is possible iff there exists  $n \geq 2t_b + t_f + t_p + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ , where  $r \geq 2$ . From Theorem 14.12, any  $r$ -phase SSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is possible iff there exists  $n \geq t_b + \max(t_b, t_p) + t_f + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ , where  $r \geq 2$ . Therefore, except when either  $t_b = 0$  or  $t_p = 0$ , allowing a negligible error probability (only in the reliability), significantly helps in the POSSIBILITY of multiphase secure message transmission protocol.*

**Comparison 14.15 (COMPLEXITY OF MULTI PHASE SSMT AND PSMT)** *From Theorem 11.5, any multiphase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  over  $n \geq 2t_b + t_f + t_p + 1$  wires has to communicate  $\Omega\left(\frac{n\ell}{n-(2t_b+t_f+t_p)}\right)$  field elements to securely send a message containing  $\ell$  field elements. From Theorem 14.13, any multi phase SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  over  $n \geq t_b + \max(t_b, t_p) + t_f + 1$  wires has to communicate  $\Omega\left(\frac{n\ell}{n-(t_b+t_f+t_p)}\right)$  field elements to securely send a message containing  $\ell$  field elements.*

Let us fix  $n = 2t_b + t_f + t_p + 1$  for which both multi phase PSMT and SSMT is possible. With  $n = 2t_b + t_f + t_p + 1$ , the lower bounds for PSMT and SSMT become  $\Omega(n\ell)$  and  $\Omega\left(\frac{n\ell}{t_b}\right)$  field elements respectively. Particularly, if we consider  $\mathcal{A}_{t_b}^{static}$  then  $n$  must be at least  $2t_b + 1$  for both multi phase PSMT and SSMT to be possible. With  $n = 2t_b + 1$ , the lower bounds for PSMT and SSMT become  $\Omega(n\ell)$  and  $\Omega(\ell)$  field elements respectively, for now  $t_b = \Theta(n)$ . Hence with  $n = 2t_b + 1$  while SSMT can be achieved with constant factor overhead tolerating  $\mathcal{A}_{t_b}^{static}$ , PSMT can not be achieved with constant factor overhead tolerating  $\mathcal{A}_{t_b}^{static}$ . This shows the power of allowing a negligible error probability (only in the reliability) in multiphase secure message transmission.

#### 14.2.2.1 Existing Three Phase SSMT Protocol Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ [75]

To show the sufficiency of the condition given in Theorem 14.12, the author in [75] presented a three phase SSMT protocol with  $n = t_b + \max(t_b, t_p) + t_f + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ . The protocol securely sends a single element from  $\mathbb{F}$  by communicating  $\mathcal{O}(n^2)$  field elements, tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . We call the protocol as protocol 3-SSMT-Static-Mixed and present it in Fig. 14.2. The protocol uses certain ideas used in the SSMT protocol of [33].

The following theorem taken from [75] states that protocol 3-SSMT-Static-Mixed is an SSMT protocol.

**Theorem 14.16 ([75])** *Protocol 3-SSMT-Static-Mixed is a three phase SSMT protocol which securely sends a single field element by communicating  $\mathcal{O}(n^2)$  field elements.*

Figure 14.2: Three Phase SSMT Protocol Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ :  $n = t_b + t_f + \max(t_b, t_p) + 1, |m^S| = 1$

**Protocol 3-SSMT-Static-Mixed**

**Phase I: S to R:**

1. **S** sends to **R** two random, non-zero elements  $\rho_{i1}^S$  and  $\rho_{i2}^S$  along wire  $w_i$ , for  $i = 1, \dots, n$ .

**Phase II: R to S:**

1. Let **R** receive values along  $n' \geq (n - t_f)$  wires. **R** neglects the remaining  $(n - n')$  wires.
2. Let **R** receive  $\rho_{i1}^R$  and  $\rho_{i2}^R$  along wire  $w_i$ , if  $w_i$  is not neglected by **R**.
3. **R** chooses uniformly at random a non-zero element  $K \in \mathbb{F}$ . **R** then broadcasts to **S** the following:
  - (a) Identities of the  $(n - n')$  wires neglected by **R**;
  - (b) The random  $K$  and
  - (c) The values  $u_i = (K\rho_{i1}^R + \rho_{i2}^R)$  for all  $i$  such that  $w_i$  is not neglected by **R**.

**Phase III: S to R:**

1. **S** correctly receives the identities of the  $(n - n')$  wires neglected by **R** during **Phase II** and **S** eliminates these wires. <sup>a</sup>
2. **S** also correctly receives  $K$  and the values  $u_i$  for each  $i$ , such that wire  $w_i$  is not eliminated by **R**.
3. **S** then computes the set  $H$ , where  $H = \{w_i | u_i = (K\rho_{i1}^S + \rho_{i2}^S)\}$ .
4. **S** also computes the secret pad  $\rho^S$  where  $\rho^S = \sum_{w_i \in H} \rho_{i2}^S$ .
5. **S** then broadcasts the set  $H$  and the blinded message  $Z = m^S \oplus \rho^S$  to **R**, where  $m^S$  is the secret message, which **S** wants to send securely to **R**.

**Message Recovery by R**

1. **R** correctly receives  $H$  and computes his version of  $\rho^R$  (which is equal to  $\rho^S$  with very high probability).
2. **R** correctly receives the blinded message  $Z$  and outputs  $m^R = Z \oplus \rho^R$ .

<sup>a</sup> Irrespective of the values of  $t_b$  and  $t_p$ ,  $n$  is at least  $2t_b + t_f + 1$  and any information which is broadcast over these many wires will be received correctly by taking the majority.

From Theorem 14.13 and Theorem 14.16, we find that protocol 3-SSMT-Static-Mixed is not a communication optimal SSMT protocol. To the best of our knowledge, there exists no multi phase communication optimal SSMT protocol in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . So in the next section, we present a new multiphase communication optimal SSMT protocol in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . This will completely resolve the issue of OPTIMALITY of multi phase SSMT in undirected synchronous networks.

### 14.3 A Four Phase Communication Optimal SSMT

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = t_b + \max(t_b, t_p) + t_f + 1$  wires. We then design a four phase communication optimal SSMT protocol called 4-Optimal-SSMT-Static-Mixed, tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . The total communication complexity of the protocol is  $\mathcal{O}(n^3)$  field elements. If  $t_b \leq t_p$ , then the protocol securely sends a message  $m^{\mathbf{S}}$  containing  $n^2$  field elements. On the other hand, if  $t_p > t_b$  then the protocol securely sends a message  $m^{\mathbf{S}}$  containing  $(t_p - t_b)n^2$  field elements. Protocol 4-Optimal-SSMT-Static-Mixed uses three phase SSMT protocol 3-SSMT-Static-Mixed (see Fig. 14.2) and single phase SRMT protocol 1-Optimal-SRMT-Static-Mixed (see Fig. 8.3) as a black box.

The high level idea of protocol 4-Optimal-SSMT-Static-Mixed is as follows: During first phase,  $\mathbf{R}$  sends a random  $n^2$ -tuple to  $\mathbf{S}$  over each wire.  $\mathbf{R}$  also hashes each  $n^2$ -tuple by a random hash key.  $\mathbf{R}$  then securely sends all the hash keys and hash values by using the three phase SSMT protocol 3-SSMT-Static-Mixed. Now using the hash keys and hash values,  $\mathbf{S}$  can find out the wires who have correctly delivered the  $n^2$ -tuples, sent by  $\mathbf{R}$ .

If  $t_b \leq t_p$ , then  $n = t_b + t_p + t_f + 1$  and so there exists at least one honest wire, who will correctly and securely deliver the  $n^2$ -tuple to  $\mathbf{S}$ . On the other hand, if  $t_b > t_p$ , then there exists at least  $(t_b - t_p)$  honest wires, who will correctly and securely deliver the  $n^2$ -tuple to  $\mathbf{S}$ . Accordingly,  $\mathbf{S}$  applies EXTRAND on the  $n^2$  tuples which are correctly received by  $\mathbf{S}$  and generates a pad of size either  $n^2$  field elements or  $(t_b - t_p)n^2$  field elements. Accordingly,  $\mathbf{S}$  takes a message of size either  $n^2$  or  $(t_b - t_p)n^2$  and mask the message using the pad. If the masked message is of size  $n^2$ , then  $\mathbf{S}$  reliably sends it by broadcasting it to  $\mathbf{R}$ . Otherwise  $\mathbf{S}$  reliably sends the masked message by invoking single phase SRMT protocol 1-Optimal-SRMT-Static-Mixed. Protocol 4-Optimal-SSMT-Static-Mixed is now formally presented in Fig. 14.4.

We now prove the properties of protocol 4-Optimal-SSMT-Static-Mixed.

**Claim 14.17** *In protocol 4-Optimal-SSMT-Static-Mixed,  $\mathbf{S}$  will correctly receive each  $\rho_i^{\mathbf{R}}$  and  $y_i^{\mathbf{R}}$ , except with error probability  $2^{-\Omega(\kappa)}$ . Moreover, the adversary will have no information about these values.*

PROOF: The proof follows from the fact that in protocol 4-Optimal-SSMT-Static-Mixed,  $\mathbf{R}$  sends each  $\rho_i^{\mathbf{R}}$  and  $y_i^{\mathbf{R}}$  by executing three phase SSMT protocol 3-SSMT-Static-Mixed. The rest now follows from the properties of protocol 3-SSMT-Static-Mixed (see Theorem 14.16).  $\square$

**Claim 14.18** *In protocol 4-Optimal-SSMT-Static-Mixed, if  $w_i$  is corrupted and delivers incorrect  $n^2$ -tuple, then except with error probability  $2^{-\Omega(\kappa)}$ ,  $\mathbf{S}$  will include wire  $w_i$  in  $L_{fault}$ .*

PROOF: From the previous claim,  $\rho_i^{\mathbf{S}} = \rho_i^{\mathbf{R}}$  and  $y_i^{\mathbf{S}} = y_i^{\mathbf{R}}$ , except with error probability  $2^{-\Omega(\kappa)}$ . Moreover, adversary will have no information about the hash value and hash

key. Now suppose that wire  $w_i$  is corrupted and delivers incorrect  $n^2$ -tuple. Then from the property of hashing,  $\mathbf{S}$  will be able to detect it, except with error probability  $\frac{n^2}{|\mathbb{F}|} \approx 2^{-\Omega(\kappa)}$ . So except with error probability  $2^{-\Omega(\kappa)}$ ,  $\mathbf{S}$  will include wire  $w_i$  in  $L_{\text{fault}}$ .  $\square$

**Claim 14.19** *In protocol 4-Optimal-SSMT-Static-Mixed, if the masked message  $d$  is broadcast then it will be delivered correctly without any error. On the other hand, if the masked message  $d$  is sent by executing protocol 1-Optimal-SRMT-Static-Mixed, then it will be delivered correctly, except with error probability  $2^{-\Omega(\kappa)}$ .*

PROOF: Notice that irrespective of the value of  $t_b$  and  $t_p$ ,  $n \geq 2t_b + t_f + 1$  and so any value broadcasted over  $n$  wires will be delivered correctly. Thus, if  $d$  is broadcasted then it will be delivered correctly. On the other hand, if  $d$  is sent by executing protocol 1-Optimal-SRMT-Static-Mixed, then by the property of 1-Optimal-SRMT-Static-Mixed, it will be delivered correctly, except with error probability  $2^{-\Omega(\kappa)}$ .  $\square$

**Lemma 14.20** *In protocol 4-Optimal-SSMT-Static-Mixed, the message  $m^{\mathbf{S}}$  will be information theoretically secure.*

PROOF: First of all by the properties of 3-SSMT-Static-Mixed,  $\rho_i^{\mathbf{R}}$ 's and  $y_i^{\mathbf{R}}$ 's will be information theoretically secure. The proof is now divided into the following two cases:

1. **Case I: If  $t_p \geq t_b$ :** In this case,  $n = t_b + t_p + t_f + 1$ . In the worst case, the adversary can passively listen the contents over  $t_b + t_p$  wires and block  $t_f$  wires. So there will be only one honest wire  $w_i$  and hence the adversary will have no information about the  $n^2$ -tuple sent over  $w_i$ . In this case,  $\mathbf{S}$  generates a random pad of length  $n^2$  and sends  $m^{\mathbf{S}}$  containing  $n^2$  field elements, using this pad. Now the proof follows from the secrecy property of EXTRAND and working of the protocol.
2. **Case II: If  $t_b > t_p$ :** In this case,  $n = 2t_b + t_f + 1$ . In the worst case, the adversary can passively listen the contents of at most  $t_b + t_p$  wires and block  $t_f$  wires. So there will be  $(t_b - t_p)$  wires which will not be under the control of the adversary and hence the adversary will have no information about the  $n^2$ -tuples sent over these wires. In this case,  $\mathbf{S}$  generates a random pad of length  $(t_b - t_p)n^2$  and sends  $m^{\mathbf{S}}$  containing  $(t_b - t_p)n^2$  field elements, using this pad. Now the proof follows from the secrecy property of EXTRAND and working of the protocol.  $\square$

**Lemma 14.21** *The communication complexity of protocol 4-Optimal-SSMT-Static-Mixed is  $\mathcal{O}(n^3)$  field elements.*

PROOF: During **Phase I**,  $\mathbf{R}$  sends  $n^2$  random field elements over each of the  $n$  wires causing a communication complexity of  $\mathcal{O}(n^3)$  field elements.  $\mathbf{R}$  also invokes  $2n$  parallel executions of protocol 3-SSMT-Static-Mixed, each having a communication complexity of  $\mathcal{O}(n^2)$  field elements (see Theorem 14.16). This incurs total communication cost of  $\mathcal{O}(n^3)$  field elements. During **Phase IV**,  $\mathbf{S}$  sends  $d$  to  $\mathbf{R}$ . If  $t_p \geq t_b$ , then  $d$  will consist of  $n^2$  field elements and hence broadcasting it to  $\mathbf{R}$  incurs a communication complexity of  $\mathcal{O}(n^3)$ . On the other hand, if  $t_b > t_p$ , then  $d$  consist of  $(t_b - t_p)n^2$  field elements. In this case,  $\mathbf{S}$  will send  $d$  by invoking  $\frac{(t_b - t_p)}{t_b} \times n$  parallel executions of single phase SRMT protocol. Since, each execution of the single phase SRMT protocol has a communication complexity of  $\mathcal{O}(n^2)$  field elements (see Lemma 8.14), total communication complexity for sending  $d$  will be  $\mathcal{O}\left(\frac{(t_b - t_p)n^3}{t_b}\right)$ , which is  $\mathcal{O}(n^3)$ . Thus, overall communication complexity of protocol 4-Optimal-SSMT-Static-Mixed is  $\mathcal{O}(n^3)$  field elements.  $\square$

**Theorem 14.22** *Protocol 4-Optimal-SSMT-Static-Mixed is a four phase communication optimal SSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .*

PROOF: The proof that protocol 4-Optimal-SSMT-Static-Mixed is a four phase SSMT protocol follows from the protocol steps and Claim 14.17, Claim 14.18, Claim 14.19 and Lemma 14.20. We now show that the protocol is communication optimal.

Protocol 4-Optimal-SSMT-Static-Mixed sends  $(t_b - t_p)n^2\kappa$  bits (if  $t_b > t_p$ ) or  $n^2\kappa$  bits (if  $t_b \leq t_p$ ), by communicating  $O(n^3\kappa)$  bits, where  $n = t_b + \max(t_b, t_p) + t_o + t_f + 1$ . From Theorem 14.13, if  $t_b \geq t_p$  (in this case  $n = 2t_b + t_f + 1$ ), then any four phase SSMT protocol needs to communicate  $\Omega(n^3\kappa)$  bits to securely send  $(t_b - t_p)n^2\kappa$  bits. Similarly, if  $t_p > t_b$  (in this case,  $n = t_b + t_p + t_f + 1$ ), then any four phase SSMT protocol needs to communicate  $\Omega(n^3\kappa)$  bits in order to securely send  $n^2\kappa$  bits. Since the total communication complexity of 4-Optimal-SSMT-Static-Mixed in both the cases is  $O(n^3\kappa)$  bits, the protocol is communication optimal.  $\square$

### 14.3.1 SSMT with Constant Factor Overhead Tolerating $\mathcal{A}_{t_b}^{static}$

From Theorem 14.22, we get the following corollary:

**Corollary 14.22.1** *If protocol 4-Optimal-SSMT-Static-Mixed is executed only in the presence of Byzantine adversary  $\mathcal{A}_{t_b}^{static}$  (i.e.,  $t_f = t_p = 0$ ), then it achieves security with “constant factor overhead” in four phases by securely sending  $\Theta(n^3)$  field elements with a communication complexity of  $\mathcal{O}(n^3)$  field elements.*

PROOF: If protocol 4-Optimal-SSMT-Static-Mixed is executed only in the presence of  $\mathcal{A}_{t_b}^{static}$  (i.e.,  $t_f = t_p = 0$ ) then it sends  $t_b n^2 = \Theta(n^3)$  field elements in four phases by communicating  $\mathcal{O}(n^3)$  field elements (if  $t_f = t_p = 0$ , then  $n = 2t_b + 1$  and so  $t_b = \Theta(n)$ ). Thus we get *secrecy* with *constant* factor overhead in four phases when 4-Optimal-SSMT-Static-Mixed is executed under the presence of  $\mathcal{A}_{t_b}^{static}$ .  $\square$

## 14.4 Concluding Remarks and Open Problems

In this chapter, we presented a four phase communication optimal SSMT protocol in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . This, along with Theorem 14.12 and Theorem 14.13 completely settles the issue of POSSIBILITY, FEASIBILITY and OPTIMALITY of SSMT in undirected synchronous network tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . The results for SSMT in undirected synchronous network are summarized in Fig. 14.3.

Figure 14.3: Summary of the Results for SSMT in Undirected Synchronous Network Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

Number of Phases ( $r$ )	Connectivity Requirement ( $n$ )	Lower Bound on Communication Complexity	Upper Bound
$r = 1$	$n \geq 2t_b + t_f + t_p + 1$ Theorem 14.2	$\Omega\left(\frac{n\ell}{n-(2t_b+t_f+t_p)}\right)$ Theorem 14.3	Theorem 14.11
$r \geq 2$	$n \geq t_b + t_f + \max(t_b, t_p) + 1$ Theorem 14.12	$\Omega\left(\frac{n\ell}{n-(t_b+t_f+t_p)}\right)$ Theorem 14.13	Theorem 14.22

This chapter leaves two open problems:

**Open Problem 15** *Protocols 4-Optimal-SSMT-Static-Mixed and 1-Optimal-SSMT-Static-Mixed are asymptotically communication optimal. It would be interesting to design communication optimal SSMT protocols which are optimal for messages of any size.*

**Open Problem 16** *Protocol 4-Optimal-SSMT-Static-Mixed takes four phases. However, the lower bound given in Theorem 14.13 holds for any multi phase SSMT protocol. So it is an interesting open problem to reduce the phase complexity of protocol 4-Optimal-SSMT-Static-Mixed, while keeping the communication complexity same. In general, it is an interesting open problem to come up with an efficient two phase communication optimal SSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .*

Figure 14.4: Four Phase Communication Optimal SSMT Protocol Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ :  
 $n = t_b + t_f + \max(t_b, t_p) + 1$ . If  $t_b \leq t_p$ , then  $|m^S| = n^2$ , else  $|m^S| = (t_b - t_p)n^2$

**Protocol 4-Optimal-SSMT-Static-Mixed**

**Phase I: R to S:**

1. For  $i = 1, \dots, n$ , **R** selects a random, non-zero  $n^2$ -tuple  $(r_{i,1}^R, \dots, r_{i,n^2}^R)$ .
2. For  $i = 1, \dots, n$ , corresponding to wire  $w_i$ , **R** selects a random, non-zero hash key  $\rho_i^R$ .
3. For  $i = 1, \dots, n$ , **R** computes  $y_i^R = \text{hash}(\rho_i^R; r_{i,1}^R, \dots, r_{i,n^2}^R)$ .
4. For  $i = 1, \dots, n$ , **R** sends  $(r_{i,1}^R, \dots, r_{i,n^2}^R)$  to **S** over wire  $w_i$ .
5. In addition to the above communication, **R** executes  $2n$  instances of 3-SSMT-Static-Mixed to securely send  $\rho_1^R, \dots, \rho_n^R, y_1^R, \dots, y_n^R$ . This will occupy **Phase I, II** and **III**.<sup>a</sup>

The instances of 3-SSMT-Static-Mixed, executed during **Phase I** will terminate at the end of **Phase III**.

**Phase IV: S to R:**

1. Let  $\mathcal{F}$  denote the set of wires which delivered no information to **S**. Let **S** receive the  $n^2$ -tuple  $(r_{i,1}^S, \dots, r_{i,n^2}^S)$  over wire  $w_i \notin \mathcal{F}$ .
2. Let **S** receive  $\rho_1^S, \dots, \rho_n^S, y_1^S, \dots, y_n^S$  at the end of  $2n$  instances of 3-SSMT-Static-Mixed.
3. For every  $w_i \notin \mathcal{F}$ , **S** verifies whether  $y_i^S \stackrel{?}{=} \text{hash}(\rho_i^S; r_{i,1}^S, \dots, r_{i,n^2}^S)$ . If the test fails, then **S** adds  $w_i$  to a list  $L_{\text{fault}}$ .
4. **S** sets  $L_{\text{honest}} = \mathcal{W} \setminus (L_{\text{fault}} \cup \mathcal{F})$ . Here  $\mathcal{W}$  is the set of  $n$  wires  $w_1, \dots, w_n$ .
5. If  $t_p \geq t_b$ , then **S** computes a random pad  $Z^S$  of size  $n^2$  field elements from the  $n^2|L_{\text{honest}}|$  field elements which are received over the wires in  $L_{\text{honest}}$  as follows:
$$Z^S = \text{EXTRAND}_{n^2|L_{\text{honest}}|, n^2}(r_{i,j}^S | w_i \in L_{\text{honest}}, 1 \leq j \leq n^2).$$
6. If  $t_b > t_p$ , then **S** computes a random pad  $Z^S$  of size  $(t_b - t_p)n^2$  as follows:
$$Z^S = \text{EXTRAND}_{n^2|L_{\text{honest}}|, (t_b - t_p)n^2}(r_{i,j}^S | w_i \in L_{\text{honest}}, 1 \leq j \leq n^2).$$
7. **S** computes  $d = m^S \oplus Z^S$ .
8. If  $t_p \geq t_b$  then  $d$  is of size  $n^2$ , so **S** broadcasts  $d$  to **R**.
9. If  $t_b > t_p$  then  $d$  consists of  $(t_b - t_p)n^2$  field elements. In this case, **S** reliably sends  $d$  to **R** by invoking  $\frac{(t_b - t_p)}{t_b} \times n$  parallel executions of 1-Optimal-SRMT-Static-Mixed.<sup>b</sup>
10. **S** also broadcasts the set  $L_{\text{fault}}$  and  $\mathcal{F}$  to **R**.

**Message recovery by R**

1. **R** correctly receives  $L_{\text{fault}}$  and  $\mathcal{F}$  and sets  $L_{\text{honest}} = \mathcal{W} \setminus (L_{\text{fault}} \cup \mathcal{F})$ .
2. **R** correctly receives  $d$  with certainty (probability one) when  $t_p \geq t_b$  and with high probability when  $t_b > t_p$ .
3. If  $t_b \leq t_p$ , then **R** computes  $Z^R$  of size  $n^2$  field elements as follows:
$$Z^R = \text{EXTRAND}_{n^2|L_{\text{honest}}|, n^2}(r_{i,j}^R | w_i \in L_{\text{honest}}, 1 \leq j \leq n^2).$$
4. If  $t_b > t_p$ , then **R** computes  $Z^R$  of size  $(t_b - t_p)n^2$  field elements as follows:
$$Z^R = \text{EXTRAND}_{n^2|L_{\text{honest}}|, (t_b - t_p)n^2}(r_{i,j}^R | w_i \in L_{\text{honest}}, 1 \leq j \leq n^2).$$
5. Once  $Z^R$  is computed, **R** recovers  $m^R$  by computing  $m^R = Z^R \oplus d$ .

<sup>a</sup> Recall that in a single instance of 3-SSMT-Static-Mixed, only one element can be sent securely.

<sup>b</sup> This is possible because  $n$  is at least  $2t_b + t_f + 1$ , which is sufficient for single phase SRMT. Since a single instance of 1-Optimal-SRMT-Static-Mixed reliably sends  $nt_b$  field elements, vector  $d$  consisting of  $(t_b - t_p)n^2$  field elements can be communicated by **S** by invoking the single phase SRMT protocol  $\frac{(t_b - t_p)}{t_b} \times n$  times parallelly.

## Chapter 15

# SRMT and SSMT in Directed Networks Tolerating Static Byzantine Adversary

The SRMT and SSMT problem in directed network was first introduced in [24], where the authors have given the necessary and sufficient condition for SRMT and SSMT in directed network tolerating  $\mathcal{A}_{t_b}^{static}$ . Desmedt et al. [24, 87] also presented an SSMT protocol, satisfying their characterization. The authors in [24, 87] claimed that their protocol is *efficient* and has polynomial computational and communication complexity. However, in this chapter, we show that it is not so. That is, we specify an adversary strategy, which may cause the protocol to have exponential computational and communication complexity<sup>1</sup>. We then present new and efficient SRMT and SSMT protocols, satisfying the characterization of [24, 87]. Finally we show that our proposed protocols are communication optimal by deriving lower bound on the communication complexity of SRMT and SSMT protocols in directed network. To the best of our knowledge, our protocols are the first communication optimal SRMT and SSMT protocols in directed networks.

We now discuss the network model and adversary settings used in this chapter.

### 15.1 Network Model and Adversary Settings

The network model and adversary settings used in this chapter are same as in Chapter 7. Specifically, we assume that there are  $n$  wires directed from  $\mathbf{S}$  to  $\mathbf{R}$ , denoted by  $f_1, \dots, f_n$ , which are also called as *top band*. Moreover, we assume that there are  $u$  wires directed from  $\mathbf{R}$  to  $\mathbf{S}$ , denoted by  $b_1, \dots, b_u$ , which are also called as *bottom band*. Furthermore, we assume that the wires in the top band are node disjoint from the wires in the bottom band. We assume the presence of a computationally unbounded static Byzantine adversary  $\mathcal{A}_{t_b}^{static}$ , who can corrupt at most  $t_b$  wires, out of  $n + u$  wires. We assume that  $n = \max(2t_b - u + 1, t_b + 1)$ ,  $u \leq t_b$  and  $n + u = 2t_b + 1$ . From Theorem 15.1, this is the minimum number of wires required for the possibility of SRMT and SSMT in directed network, tolerating  $\mathcal{A}_{t_b}^{static}$ .

To bound the error probability<sup>2</sup> of our SRMT and SSMT protocols by  $2^{-\Omega(\kappa)}$ , we assume that all computation and communication is done over a finite field  $\mathbb{F}$ , where

---

<sup>1</sup>In fact, this is applicable for the SRMT and SSMT protocols presented in [54].

<sup>2</sup>Recall that in SRMT and SSMT,  $\mathbf{R}$  should output the correct message, except with error probability  $2^{-\Omega(\kappa)}$ .

$\mathbb{F} = GF(2^\kappa)$ . Here  $\kappa$  is the error parameter. Thus each field element can be represented by  $\mathcal{O}(\kappa)$  bits. Moreover, without loss of generality, we assume that  $n = \text{poly}(\kappa)$ .

### 15.1.1 Comparison of Our Model with the Model of [84, 72, 80]

In this thesis, for directed networks, we considered "wired" network model, where it is assumed that the intermediary nodes between  $\mathbf{S}$  and  $\mathbf{R}$  are just message forwarding nodes. Under this assumption, we abstract the network in the form of directed wires, directed either from  $\mathbf{S}$  to  $\mathbf{R}$  or  $\mathbf{R}$  to  $\mathbf{S}$ . The same abstraction is done in [24]. However, in an *arbitrary directed* network, if the intermediate nodes (other than  $\mathbf{S}$  and  $\mathbf{R}$ ) are allowed to carry out computation and communication (beyond just acting as a message forwarding node), as in the case of a *virtual private network (VPN)*, then the wired abstraction results in loss of generality. The insufficiency of wired abstraction in such a network model is pointed out in [84, 72, 80] where characterizations for SRMT and SSMT over the arbitrary network, treating entire graph in its full form are also reported. While authors in [72] have considered threshold adversary, the authors of [84, 80] have considered non-threshold<sup>3</sup> adversary for their characterization. However, it is likely to take exponential time to verify whether a given arbitrary directed network satisfies the characterization given in [84, 72, 80] for the possibility of SRMT and SSMT. Moreover, the protocols given in [84, 72, 80] require exponential computational and communication complexity and are highly non-intuitive.

It should be noted that abstracting the underlying network to a bunch of wires is *incomparable* to treating the network in its full form. Both are sensible and practical. We need to decide which model to follow based on the characteristic of given underlying network. In this thesis, we considered the wire model, as it is relatively simple. Moreover, it is relatively easy to design protocols in wire model, in comparison to design protocols by considering the graph in its entirety.

We now present the existing results for SRMT and SSMT in directed synchronous networks tolerating  $\mathcal{A}_{t_b}^{static}$ .

## 15.2 Existing Results for SRMT and SSMT in Directed Network

As mentioned earlier, SRMT and SSMT in directed network was first studied by Desmedt et al. [24]. Specifically, they gave the following characterization:

**Theorem 15.1** ([24, 87]) *Suppose there exists  $u \leq t_b$  wires in the bottom band and  $n$  wires in the top band, such that the wires in the top band are disjoint from the wires in the bottom band. Then any SRMT/SSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff  $n = \max(2t_b - u + 1, t_b + 1)$ .*

In [87], the authors presented an SSMT protocol, satisfying the above characterization. Moreover, the authors claimed that their protocol is efficient (see Theorem 3.4 of [87]). To the best of our knowledge, these are the only results for SRMT and SSMT in directed networks.

---

<sup>3</sup>A non-threshold adversary is a generalized form of the threshold adversary.

## 15.3 Overview of Our Results for SRMT and SSMT in Directed Network

We now summarize the contributions of this chapter:

1. We first show that the SSMT protocol presented in [24, 87] is inefficient. We do this by specifying an adversary strategy, which may cause the protocol to have exponential computational and communication complexity. In fact, we show that the same adversary strategy is applicable for the SRMT and SSMT protocols presented in [54], thus making the SRMT and SSMT protocols of [54] to have exponential computational and communication complexity.
2. We then present new and efficient SRMT and SSMT protocols. These protocols are achieved by making several modifications to the SRMT and SSMT protocols presented in [54]. In fact, these modifications when applied to the SSMT protocol of [24, 87] will make it efficient. In short, our SRMT and SSMT protocols have the following properties:
  - (a) Our SRMT protocol takes  $\mathcal{O}(u)$  phases and reliably sends a message containing  $\Theta(n^3\kappa)$  bits by overall communicating  $\mathcal{O}(n^3\kappa)$  bits. Thus, our SRMT protocol achieves reliability with constant factor overhead.
  - (b) Our SSMT protocol takes  $\mathcal{O}(u)$  phases and has a communication complexity of  $\mathcal{O}(n^3\kappa)$  bits. If the entire bottom band is corrupted, then the protocol securely sends a message containing  $\Theta(n^2u\kappa)$  bits. Otherwise, the protocol securely sends a message containing  $\Theta(n^2\kappa)$  bits.
3. Finally, we show that our SRMT and SSMT protocols are asymptotically communication optimal. For this, we derive the lower bound on the communication complexity of SRMT and SSMT protocols in directed networks. Specifically, we show the following:
  - (a) Any SRMT protocol must communicate  $\Omega(\ell\kappa)$  bits to reliably send a message containing  $\ell\kappa$  bits.
  - (b) If the entire bottom band is corrupted then any SSMT protocol must communicate  $\Omega(\frac{n\ell}{u}\kappa)$  bits to securely send a message containing  $\ell\kappa$  bits. On the other hand, if the entire bottom band is not corrupted then any SSMT protocol must communicate  $\Omega(n\ell\kappa)$  bits to securely send a message containing  $\ell\kappa$  bits.

### 15.3.1 Overview of Our Protocols

We first design a three phase SSMT protocol called 3-SSMT-Static-Byzantine-Directed, which sends a message containing  $\Theta(\frac{n}{3})$  field elements by communicating  $\mathcal{O}(n^3)$  field elements. Then using this protocol as a black-box, we design a six phase protocol called 6-Pad, which securely establishes a random, one time pad between  $\mathbf{S}$  and  $\mathbf{R}$ . Then using protocol 6-Pad as a black-box, we design our communication optimal SRMT protocol called  $u$ -Optimal-SRMT-Static-Byzantine-Directed, which takes  $\mathcal{O}(u)$  phases. Finally, using protocol 6-Pad and protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed as a black-box, we design our  $\mathcal{O}(u)$  phase communication optimal SSMT protocol called  $u$ -Optimal-SSMT-Static-Byzantine-Directed.

The idea behind protocol *u-Optimal-SSMT-Static-Byzantine-Directed* is as follows: we first execute protocol *6-Pad* to securely establish a one time pad between **S** and **R**. Then using this pad, **S** masks the secret message and reliably sends the masked message by using the protocol *u-Optimal-SRMT-Static-Byzantine-Directed*. After receiving the masked message, **R** can unmask the message by using the pad.

### 15.3.2 The Roadmap

The chapter is organized as follows: in the next section, we specify the tools that are used in our protocols. In Section 15.5, we present our three phase SSMT protocol called *3-SSMT-Static-Byzantine-Directed*. In Section 15.5.3, we show the inefficiency of the SRMT and SSMT protocols of [24, 54, 87]. Section 15.6 presents our six phase pad establishment protocol *6-Pad*. Our communication optimal SRMT protocol is presented in Section 15.7. In Section 15.8, we present our communication optimal SSMT protocol. The lower bounds on the communication complexity of SRMT and SSMT protocols are presented in Section 15.9. The chapter ends with a conclusion and directions for further research.

## 15.4 Tools Used in Our SRMT and SSMT Protocols

To design our SRMT and SSMT protocols, we use the following tools in this chapter:

1. Unconditionally Reliable Authentication: see Definition 8.1.
2. Unconditionally Secure Authentication: see Definition 8.2.
3. Unconditional Hashing: see Definition 8.3.
4. Extracting Randomness: see Fig. 9.2.
5. Extrapolation Technique: see Section 8.2.2.

## 15.5 A Three Phase SSMT Protocol in Directed Network Tolerating $\mathcal{A}_{t_b}^{static}$

We now present a three phase SSMT protocol called *3-SSMT-Static-Byzantine-Directed*. The protocol securely sends a message  $m^{\mathbf{S}}$  containing  $\frac{n}{3}$  field elements by communicating  $\mathcal{O}(n^3)$  field elements. The protocol will be later used in our communication optimal SRMT and SSMT protocol. The protocol uses certain ideas from the SSMT protocol of [24]. In addition, the protocol also uses certain new ideas proposed by us. Before presenting protocol *3-SSMT-Static-Byzantine-Directed*, we present another three phase SSMT protocol called *3-SSMT-Static-Byzantine-Directed-Exponential*, which requires exponential communication and computational complexity. The main reason for presenting *3-SSMT-Static-Byzantine-Directed-Exponential* is to present the underlying principle, which we try to simulate in protocol *3-SSMT-Static-Byzantine-Directed*, while maintaining polynomial computation and communication complexity. A protocol somewhat similar to protocol *3-SSMT-Static-Byzantine-Directed-Exponential* is also presented in [24]. So in the next section, we first present protocol *3-SSMT-Static-Byzantine-Directed-Exponential*.

### 15.5.1 A Three Phase Exponential SSMT Protocol

Let  $\mathcal{P}_1, \dots, \mathcal{P}_k$  denote the enumeration of all possible  $t_b + 1$ -sized subset of the wire set  $\{f_1, \dots, f_n, b_1, \dots, b_u\}$ . Thus  $k = \binom{2t_b+1}{t_b+1}$ , as  $n + u = 2t_b + 1$ . Since there are at least  $t_b + 1$  honest wires including top and bottom band, there exists at least one path set, say  $\mathcal{P}_i$ , where  $\mathcal{P}_i$  contains all  $t_b + 1$  honest wires. Moreover, each path set  $\mathcal{P}_j$  will have at least one wire from top band, as there can be at most  $t_b$  wires in the bottom band. If somehow  $\mathbf{S}$  and  $\mathbf{R}$  comes to know the identity of the honest path set  $\mathcal{P}_i$ , which consists of only honest wires, then  $\mathbf{S}$  and  $\mathbf{R}$  can share an  $n$ -tuple over each wire in  $\mathcal{P}_i$ . Then adding all such  $n$ -tuples,  $\mathbf{S}$  and  $\mathbf{R}$  can agree on an  $n$ -tuple, about which adversary will have no information. Finally, using the elements of the resultant  $n$ -tuple as encryption and authentication keys,  $\mathbf{S}$  can reliably and securely send  $m^{\mathbf{S}}$  over all the wires in the top band, which are present in  $\mathcal{P}_i$ .

Since, neither  $\mathbf{S}$  nor  $\mathbf{R}$  will know the exact identity of honest path set  $\mathcal{P}_i$  in advance, they have to parallelly do the above procedure for all paths sets  $\mathcal{P}_1, \dots, \mathcal{P}_k$ . During this process, if the adversary tries to change the information over the wires in any path set then with very high probability,  $\mathbf{R}$  will detect this and will neglect the information which is exchanged over the wires in that path set. The protocol is given in Fig. 15.1.

We now prove the properties of protocol 3-SSMT-Static-Byzantine-Directed-Exponential.

**Lemma 15.2** *In protocol 3-SSMT-Static-Byzantine-Directed-Exponential, adversary will have no information about  $m^{\mathbf{S}}$ .*

PROOF: Every path set  $\mathcal{P}_m$  will have at least one honest wire, either in the top band or in the bottom band. So the adversary will have no information about the  $n$ -tuple which is exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  over that wire. As a result, the adversary will have no information about the keys used by  $\mathbf{S}$ , corresponding to the path set  $\mathcal{P}_m$ . The rest now follows from the properties of *USauth*.  $\square$

**Lemma 15.3** *In protocol 3-SSMT-Static-Byzantine-Directed-Exponential,  $\mathbf{R}$  will always terminate.*

PROOF: The proof follows from the fact that there exists at least one path set  $\mathcal{P}_m$ , which will contain only honest wires.  $\square$

**Lemma 15.4** *In protocol 3-SSMT-Static-Byzantine-Directed-Exponential, if  $\mathbf{R}$  outputs  $m^{\mathbf{R}}$  then except with error probability  $2^{-\Omega(\kappa)}$ ,  $m^{\mathbf{R}} = m^{\mathbf{S}}$ .*

PROOF: Suppose  $\mathbf{R}$  outputs  $m^{\mathbf{R}}$  from  $\mathcal{S}_m^{\mathbf{R}}$ , corresponding to path set  $\mathcal{P}_m$ . Since there is at least one honest wire in  $\mathcal{P}_m$  and the adversary has no information about the  $n$ -tuple which is exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  over that wire, it implies that the adversary has no information about the keys computed by  $\mathbf{S}$  and  $\mathbf{R}$ , corresponding to path set  $\mathcal{P}_m$ . If the path set  $\mathcal{P}_m$  is completely honest or if the adversary is passively controlling the wires under its control in  $\mathcal{P}_m$ , then  $\mathbf{S}$  and  $\mathbf{R}$  will have the same keys and hence  $m^{\mathbf{R}} = m^{\mathbf{S}}$ . On the other hand, if the adversary has modified the tuples which are exchanged over the wires under its control in  $\mathcal{P}_m$ , then  $\mathbf{S}$  and  $\mathbf{R}$  will have different keys. But still, as explained above, the adversary will have no information about the keys. So from the properties of *USauth*, the adversary can make the verification process successful at  $\mathbf{R}$ 's end with negligible probability of  $2^{-\Omega(\kappa)}$ . Thus  $m^{\mathbf{R}} = m^{\mathbf{S}}$ , except with error probability  $2^{-\Omega(\kappa)}$ .  $\square$

**Lemma 15.5** *Protocol 3-SSMT-Static-Byzantine-Directed-Exponential requires exponential computation and communication complexity.*

Figure 15.1: Protocol 3-SSMT-Static-Byzantine-Directed-Exponential: A Three Phase Exponential SSMT Protocol,  $m^{\mathbf{S}} = \{m_1^{\mathbf{S}}, \dots, m_{\frac{n}{3}}^{\mathbf{S}}\}$ ,  $|m^{\mathbf{S}}| = \frac{n}{3}$

**Phase I: S to R:** **S** parallelly does the following computation and communication corresponding to each path set  $\mathcal{P}_m$ , for  $m = 1, \dots, k$ :

1. Corresponding to each  $f_i \in \mathcal{P}_m$ , **S** selects a random non-zero  $n$ -tuple  $(x_{1,i,m}^{\mathbf{S}}, \dots, x_{n,i,m}^{\mathbf{S}})$  and sends  $(x_{1,i,m}^{\mathbf{S}}, \dots, x_{n,i,m}^{\mathbf{S}})$  to **R** over wire  $f_i$ .

**Phase II: R to S:** **R** parallelly does the following computation and communication, corresponding to each path set  $\mathcal{P}_m$ , for  $m = 1, \dots, k$ :

1. Let **R** receive non-zero  $n$ -tuple  $(x_{1,i,m}^{\mathbf{R}}, \dots, x_{n,i,m}^{\mathbf{R}})$  from **S**, over wire  $f_i$ , corresponding to each  $f_i \in \mathcal{P}_m$ .
2. Corresponding to each  $b_i \in \mathcal{P}_m$ , **R** selects a random non-zero  $n$ -tuple  $(y_{1,i,m}^{\mathbf{R}}, \dots, y_{n,i,m}^{\mathbf{R}})$  and sends  $(y_{1,i,m}^{\mathbf{R}}, \dots, y_{n,i,m}^{\mathbf{R}})$  to **S** over wire  $b_i$ .

**Phase III: S to R:** **S** parallelly does the following computation and communication corresponding to each path set  $\mathcal{P}_m$ , for  $m = 1, \dots, k$ :

1. Let **S** receive non-zero  $n$ -tuple  $(y_{1,i,m}^{\mathbf{S}}, \dots, y_{n,i,m}^{\mathbf{S}})$  from **R**, over wire  $b_i$ , corresponding to each  $b_i \in \mathcal{P}_m$ .
2. For  $i = 1, \dots, n$ , **S** computes his version of  $n$  keys  $\mathcal{C}_{i,m}^{\mathbf{S}} = \sum_{f_j \in \mathcal{P}_m} x_{i,j,m}^{\mathbf{S}} + \sum_{b_j \in \mathcal{P}_m} y_{i,j,m}^{\mathbf{S}}$ .
3. For each element of  $m^{\mathbf{S}}$ , **S** takes three elements from the keys computed in the previous step and computes the set  $\mathcal{S}_m^{\mathbf{S}} = \{(c_{i,m}^{\mathbf{S}}, d_{i,m}^{\mathbf{S}}) : i = 1, \dots, \frac{n}{3}\}$  where  $(c_{i,m}^{\mathbf{S}}, d_{i,m}^{\mathbf{S}}) = USauth(m_i^{\mathbf{S}}; \mathcal{C}_{3i-2,m}^{\mathbf{S}}, \mathcal{C}_{3i-1,m}^{\mathbf{S}}, \mathcal{C}_{3i,m}^{\mathbf{S}})$ , for  $i = 1, \dots, \frac{n}{3}$ .
4. **S** sends the set  $\mathcal{S}_m^{\mathbf{S}}$  to **R** over all the top band wires in the set  $\mathcal{P}_m$  and terminates the protocol.

**Message Recovery by R:** If **R** receives  $\mathcal{S}_m^{\mathbf{R}} = \{(c_{i,m}^{\mathbf{R}}, d_{i,m}^{\mathbf{R}}) : i = 1, \dots, \frac{n}{3}\}$  over all  $f_j$ 's in path set  $\mathcal{P}_m$ , then corresponding to path set  $\mathcal{P}_m$ , **R** does the following computation:

1. For  $i = 1, \dots, n$ , **R** computes his version of  $n$  keys  $\mathcal{C}_{i,m}^{\mathbf{R}} = \sum_{f_j \in \mathcal{P}_m} x_{i,j,m}^{\mathbf{R}} + \sum_{b_j \in \mathcal{P}_m} y_{i,j,m}^{\mathbf{R}}$ .
2. For  $i = 1, \dots, \frac{n}{3}$ , **R** checks whether  $d_{i,m}^{\mathbf{R}} \stackrel{?}{=} \mathcal{C}_{3i-1,m}^{\mathbf{R}} c_{i,m}^{\mathbf{R}} + \mathcal{C}_{3i,m}^{\mathbf{R}}$ .
3. If the above test passes for all  $i = 1, \dots, \frac{n}{3}$ , then **R** computes  $m_i^{\mathbf{R}} = c_{i,m}^{\mathbf{R}} - \mathcal{C}_{3i-2,m}^{\mathbf{R}}$ . **R** then concatenates  $m_1^{\mathbf{R}}, \dots, m_{\frac{n}{3}}^{\mathbf{R}}$  to recover  $m^{\mathbf{R}}$  and terminates.

We now proceed to the discussion of protocol 3-SSMT-Static-Byzantine-Directed, which is an efficient three phase SSMT protocol. The principle used in 3-SSMT-Static-Byzantine-Directed is similar to protocol 3-SSMT-Static-Byzantine-Directed-Exponential.

However, instead of working with all possible  $t_b + 1$ -sized subset of wires,  $\mathbf{S}$  and  $\mathbf{R}$  uses certain mechanism, which allows them to work with at most  $u$  path sets, thus making the communication and computation complexity of the protocol polynomial.

### 15.5.2 A Three Phase Efficient SSMT Protocol

In protocol 3-SSMT-Static-Byzantine-Directed,  $\mathbf{R}$  first tries to find whether there exists  $t_b + 1$  honest wires in the top band. In order to facilitate  $\mathbf{R}$  to do so,  $\mathbf{S}$  tries to send  $m^{\mathbf{S}}$  using two different methods. If there exists  $t_b + 1$  honest wires in the top band then the first method would be successful. However, if there are less than  $t_b + 1$  honest wires in the top band, then with very high probability  $\mathbf{R}$  will detect this and will conclude that at least one honest wire is present in the bottom band. So  $\mathbf{S}$  and  $\mathbf{R}$  interacts for two more phases and  $\mathbf{S}$  tries to again send  $m^{\mathbf{S}}$  using the second method in the third phase. The second method tries to follow the principle used in protocol 3-SSMT-Static-Byzantine-Directed, however ensuring that the communication and computation complexity is polynomial in  $n$ .

We now begin with the description of protocol 3-SSMT-Static-Byzantine-Directed, phase by phase. However, instead of describing the entire protocol in a single shot, we prefer to discuss each phase individually. This would help the reader to understand the nuances and ideas used in each phase. So we begin with the description of first phase of protocol 3-SSMT-Static-Byzantine-Directed, which is given in the next subsection.

#### 15.5.2.1 Phase I of Protocol 3-SSMT-Static-Byzantine-Directed

During the first phase of the protocol,  $\mathbf{S}$  tries to send  $m^{\mathbf{S}}$  using the first method.  $\mathbf{S}$  also sends some additional information, which might be useful during second phase, if at all it is executed. **Phase I** of protocol 3-SSMT-Static-Byzantine-Directed is formally given in Fig. 15.2.

We now prove the properties of **Phase I** of protocol 3-SSMT-Static-Byzantine-Directed.

**Claim 15.6** *In Protocol 3-SSMT-Static-Byzantine-Directed, if  $\mathbf{R}$  concludes that  $F_i^{\mathbf{R}}$  is a valid row of  $T$ , then except with error probability  $2^{-\Omega(\kappa)}$ ,  $F_i^{\mathbf{R}} = F_i^{\mathbf{S}}$ .*

PROOF: The lemma is true without any error if wire  $f_i$  is uncorrupted. So let wire  $f_i$  be a corrupted wire, who delivers  $F_i^{\mathbf{R}} \neq F_i^{\mathbf{S}}$ . In this case, if  $F_i^{\mathbf{R}}$  is considered as a valid row of  $T$ , then it implies that  $Support_i \geq t_b + 1$ . Since there can be at most  $t_b$  corrupted wires in the top band, this implies that there exists at least one honest wire, say  $f_j$ , which correctly and securely delivered the hash key  $\alpha_j^{\mathbf{R}} = \alpha_j^{\mathbf{S}}$  and hash value  $v_{ij}^{\mathbf{R}} = v_{ij}^{\mathbf{S}} = hash(\alpha_j^{\mathbf{S}}; F_i^{\mathbf{S}}) = hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{S}})$ , such that  $f_j \in Support_i$ . Since  $f_j \in Support_i$ , it implies that  $v_{ij}^{\mathbf{R}} = hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{R}})$ . Since adversary does not know  $\alpha_j^{\mathbf{R}}$  and  $v_{ij}^{\mathbf{R}}$ , he can ensure that  $v_{ij}^{\mathbf{R}} = hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{S}})$ , as well as  $v_{ij}^{\mathbf{R}} = hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{R}})$ , where  $F_i^{\mathbf{R}} \neq F_i^{\mathbf{S}}$ , with probability at most  $\frac{n-1+t_b}{|\mathbb{F}|} \approx 2^{-\Omega(\kappa)}$ , which is negligible in our context. So with very high probability,  $f_j$  will not belong to  $Support_i$ , which is a contradiction. So with overwhelming probability  $F_i^{\mathbf{R}} = F_i^{\mathbf{S}}$ .  $\square$

**Claim 15.7** *During Phase I, at least  $n$  coefficients of  $M^{\mathbf{S}}(x)$  are information theoretically secure.*

PROOF: We consider the worst case, when  $\mathcal{A}_{t_b}^{static}$  controls at most  $t_b$  wires in the top band. Without loss of generality, let these be the first  $t_b$  wires. So  $\mathcal{A}_{t_b}^{static}$  will

Figure 15.2: Phase I of Protocol 3-SSMT-Static-Byzantine-Directed

**Phase I: S to R:** S does the following computation and communication:

1. S selects a random polynomial  $M^{\mathbf{S}}(x)$  over  $\mathbb{F}$  of degree  $n - 1 + t_b$  such that the lower order  $\frac{n}{3}$  coefficients of  $M^{\mathbf{S}}(x)$  are elements of  $m^{\mathbf{S}}$ .
2. S computes  $M^{\mathbf{S}}(1), \dots, M^{\mathbf{S}}(n + t_b)$ .
3. S then selects  $n + t_b$  random polynomials  $f_1^{\mathbf{S}}(x), \dots, f_{n+t_b}^{\mathbf{S}}(x)$  over  $\mathbb{F}$ , each of degree  $t_b$ , such that for  $i = 1, \dots, n + t_b$ ,  $f_i^{\mathbf{S}}(0) = M^{\mathbf{S}}(i)$ .
4. S evaluates each  $f_i^{\mathbf{S}}(x)$  at  $x = 1, \dots, n$  to form an  $n$ -tuple  $f_i^{\mathbf{S}} = [f_i^{\mathbf{S}}(1), \dots, f_i^{\mathbf{S}}(n)]$ .
5. S constructs an  $(n) \times (n + t_b)$  matrix  $T$  where  $i^{\text{th}}$  column of  $T$  contains the  $n$ -tuple  $f_i^{\mathbf{S}}$ , for  $i = 1, \dots, n + t_b$ . The matrix  $T$  is pictorially shown in Fig. 15.3. Let  $F_i^{\mathbf{S}} = [f_1^{\mathbf{S}}(i), \dots, f_{n+t_b}^{\mathbf{S}}(i)]$  denote the  $i^{\text{th}}$  row of  $T$ , for  $i = 1, \dots, n$ .
6. For  $i = 1, \dots, n$ , S sends the following to R along wire  $f_i$ :
  - (a) The vector  $F_i^{\mathbf{S}}$ ;
  - (b) A random non-zero hash key  $\alpha_i^{\mathbf{S}}$  and
  - (c) The  $n$ -tuple  $[v_{1i}^{\mathbf{S}}, \dots, v_{ni}^{\mathbf{S}}]$ , where for  $j = 1, \dots, n$ ,  $v_{ji}^{\mathbf{S}} = \text{hash}(\alpha_i^{\mathbf{S}}; F_j^{\mathbf{S}})$ .
7. In addition to all above computation and communication, S also selects a random non-zero  $(n + 1)$ -tuple  $(x_{1,i}^{\mathbf{S}}, \dots, x_{n+1,i}^{\mathbf{S}})$ , which is independent of  $F_i^{\mathbf{S}}$ , corresponding to every wire  $f_i$ , for  $i = 1, \dots, n$ . S then sends  $(x_{1,i}^{\mathbf{S}}, \dots, x_{n+1,i}^{\mathbf{S}})$  to R over wire  $f_i$ .

**Computation by R at the end of Phase I:**

1. Let R receive the following over wire  $f_i$ , for  $i = 1, \dots, n$ :
  - (a) The vector  $F_i^{\mathbf{R}}$ ;
  - (b) The hash key  $\alpha_i^{\mathbf{R}}$ ;
  - (c) The  $n$  tuple  $[v_{1i}^{\mathbf{R}}, \dots, v_{ni}^{\mathbf{R}}]$  and
  - (d) The  $(n + 1)$ -tuple  $(x_{1,i}^{\mathbf{R}}, \dots, x_{n+1,i}^{\mathbf{R}})$ .
2. For  $i = 1, \dots, n$ , R computes  $\text{Support}_i = |\{f_j : v_{ij}^{\mathbf{R}} = \text{hash}(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{R}})\}|$ . If  $\text{Support}_i \geq t_b + 1$ , then R concludes that  $F_i^{\mathbf{R}}$  is a valid row of  $T$ . Otherwise, R concludes that  $F_i^{\mathbf{R}}$  is an invalid row of  $T$ .
3. If R has received  $t_b + 1$  valid rows, then R reconstructs the secret  $m^{\mathbf{R}}$  from them and terminates the protocol. Otherwise, R proceeds to execute **Phase II**.

know the vectors  $F_1^{\mathbf{S}}, F_2^{\mathbf{S}}, \dots, F_{t_b}^{\mathbf{S}}$ , from which it will come to know  $t_b$  distinct points on the polynomials  $f_1^{\mathbf{S}}(x), \dots, f_{n+t_b}^{\mathbf{S}}(x)$ . But each  $f_i^{\mathbf{S}}(x)$  is of degree  $t_b$  and so  $\mathcal{A}_{t_b}^{\text{static}}$  will lack by one point to uniquely reconstruct each  $f_i^{\mathbf{S}}(x)$ . However,  $\mathcal{A}_{t_b}^{\text{static}}$  will also

Figure 15.3: Matrix  $T$  as Computed by  $\mathbf{S}$  During **Phase I** of 3-SSMT-Static-Byzantine-Directed

$M^{\mathbf{S}}(x)$ , Lower order $\frac{n}{3}$ coefficients of $M^{\mathbf{S}}(x)$ are elements of $m^{\mathbf{S}}$			
$M^{\mathbf{S}}(1)$	$M^{\mathbf{S}}(2)$	...	$M^{\mathbf{S}}(n + t_b)$
$f_1^{\mathbf{S}}(x)$	$f_2^{\mathbf{S}}(x)$	...	$f_{n+t_b}^{\mathbf{S}}(x)$
$f_1^{\mathbf{S}}(0) = M^{\mathbf{S}}(1)$	$f_2^{\mathbf{S}}(0) = M^{\mathbf{S}}(2)$	...	$f_{n+t_b}^{\mathbf{S}}(0) = M^{\mathbf{S}}(n + t_b)$
$f_1^{\mathbf{S}}(1)$	$f_2^{\mathbf{S}}(1)$	...	$f_{n+t_b}^{\mathbf{S}}(1)$
$f_1^{\mathbf{S}}(2)$	$f_2^{\mathbf{S}}(2)$	...	$f_{n+t_b}^{\mathbf{S}}(2)$
...	...	...	...
$f_1^{\mathbf{S}}(i)$	$f_2^{\mathbf{S}}(i)$	...	$f_{n+t_b}^{\mathbf{S}}(i)$
...	...	...	...
$f_1^{\mathbf{S}}(n)$	$f_2^{\mathbf{S}}(n)$	...	$f_{n+t_b}^{\mathbf{S}}(n)$

know  $t_b$  hash values corresponding to each  $F_1^{\mathbf{S}}, \dots, F_n^{\mathbf{S}}$ . Since the vectors  $F_1^{\mathbf{S}}, \dots, F_{t_b}^{\mathbf{S}}$  are already known to  $\mathcal{A}_{t_b}^{\text{static}}$ , the  $t_b$  hash values corresponding to them does not add anything new to  $\mathcal{A}_{t_b}^{\text{static}}$ 's view. Moreover, the vectors  $F_{t_b+2}^{\mathbf{S}}, \dots, F_n^{\mathbf{S}}$  can be expressed as a linear combination of vectors  $F_1^{\mathbf{S}}, \dots, F_{t_b+1}^{\mathbf{S}}$ . So the  $t_b$  hash values corresponding to  $F_{t_b+2}^{\mathbf{S}}, \dots, F_n^{\mathbf{S}}$  can always be expressed as a linear combination of the  $t_b$  hash values corresponding to  $F_1^{\mathbf{S}}, \dots, F_{t_b+1}^{\mathbf{S}}$ , which are known to the adversary. So, out of the  $t_b$  hash values corresponding to each  $F_i^{\mathbf{S}}(x), 1 \leq i \leq n$ , which are known to  $\mathcal{A}_{t_b}^{\text{static}}$ , only the  $t_b$  hash values corresponding to  $F_{t_b+1}^{\mathbf{S}}(x)$  add to  $\mathcal{A}_{t_b}^{\text{static}}$ 's view. But  $F_{t_b+1}^{\mathbf{S}}$  is of length  $n + t_b$ . So from the properties of hashing,  $(n + t_b) - t_b = n$  coefficients of  $F_{t_b+1}^{\mathbf{S}}$  will be information theoretically secure. This further implies that  $n$  coefficients of  $M^{\mathbf{S}}(x)$  are information theoretically secure.  $\square$

**Claim 15.8** *If  $\mathbf{R}$  gets  $t_b + 1$  valid rows of  $T$  then  $\mathbf{R}$  can recover  $m^{\mathbf{S}}$ .*

PROOF: If  $\mathbf{R}$  gets  $t_b + 1$  valid rows, then from them,  $\mathbf{R}$  gets  $t_b + 1$  distinct points on each  $f_i^{\mathbf{S}}(x)$ . Since each  $f_i^{\mathbf{S}}(x)$  is of degree  $t_b$ , using the  $t_b + 1$  valid rows,  $\mathbf{R}$  can reconstruct each  $f_i^{\mathbf{S}}(x)$  and hence  $f_i^{\mathbf{S}}(0) = M^{\mathbf{S}}(i)$ . Now using the  $M^{\mathbf{S}}(i)$ 's,  $\mathbf{R}$  can interpolate  $M^{\mathbf{S}}(x)$  and recover  $m^{\mathbf{S}}$ .  $\square$

**Lemma 15.9** *In protocol 3-SSMT-Static-Byzantine-Directed if  $\mathbf{R}$  recovers  $m^{\mathbf{R}}$  at the end of **Phase I**, then except with probability  $2^{-\Omega(\kappa)}$ ,  $m^{\mathbf{R}} = m^{\mathbf{S}}$ . Moreover,  $\mathcal{A}_{t_b}^{\text{static}}$  will have no information about  $m^{\mathbf{R}}$ .*

PROOF: If  $\mathbf{R}$  recovers  $m^{\mathbf{R}}$  at the end of **Phase I**, then it implies that  $\mathbf{R}$  has received  $t_b + 1$  valid rows. From Claim 15.6, all these rows are indeed the rows of  $T$  sent by  $\mathbf{S}$ , except with probability  $2^{-\Omega(\kappa)}$ . So from Claim 15.8, except with probability  $2^{-\Omega(\kappa)}$ ,  $m^{\mathbf{R}} = m^{\mathbf{S}}$ . Moreover, from Claim 15.7,  $\mathcal{A}_{t_b}^{\text{static}}$  will have no information about  $m^{\mathbf{R}}$ .  $\square$

**Lemma 15.10** *If there exists  $t_b + 1$  honest wires in the top band then  $\mathbf{R}$  will always be able to recover  $m^{\mathbf{S}}$  at the end of **Phase I** of 3-SSMT-Static-Byzantine-Directed. Otherwise, with very high probability,  $\mathbf{R}$  will detect this. During **Phase I** of 3-SSMT-Static-Byzantine-Directed,  $\mathbf{S}$  communicates  $\mathcal{O}(n^2\kappa)$  bits.*

PROOF: If there exists  $t_b + 1$  honest wires in the top band then  $\mathbf{R}$  will receive  $t_b + 1$  valid rows of  $T$  over them and hence from Claim 15.8,  $\mathbf{R}$  will correctly recover  $m^{\mathbf{R}} = m^{\mathbf{S}}$ . On the other hand if there exists less than  $t_b + 1$  honest wires in the top band then

from the proof of Claim 15.6,  $\mathbf{R}$  will receive less than  $t_b + 1$  valid rows with very high probability. So with very high probability,  $\mathbf{R}$  will detect that there are at most  $t_b$  honest wires in the top band.

During **Phase I**,  $\mathbf{S}$  sends  $\Theta(n)$  elements from  $\mathbb{F}$  over each wire. So **Phase I** requires a communication complexity of  $\mathcal{O}(n^2\kappa)$  bits.  $\square$

If  $\mathbf{R}$  is unable to recover  $m^{\mathbf{R}}$  at the end of **Phase I**, then  $\mathbf{R}$  concludes that there are at most  $t_b$  honest wires in the top band. This further implies that there exists at least one honest wire in the bottom band. So  $\mathbf{R}$  interacts with  $\mathbf{S}$  so as to enable  $\mathbf{S}$  to re-send  $m^{\mathbf{S}}$  using the second method. The next subsection describes the second phase of protocol 3-SSMT-Static-Byzantine-Directed.

### 15.5.2.2 Phase II of Protocol 3-SSMT-Static-Byzantine-Directed

If the first method to deliver  $m^{\mathbf{S}}$  fails at the end of **Phase I**, then in the second method,  $\mathbf{S}$  and  $\mathbf{R}$  interacts to securely establish a vector of length  $n$  during **Phase II**. Once this is done,  $\mathbf{S}$  can use the elements of the vector as encryption and authentication keys (as in protocol 3-SSMT-Static-Byzantine-Directed-Exponential) and using them,  $\mathbf{S}$  reliably and securely sends  $m^{\mathbf{S}}$  during **Phase III**.

Securely establishing a vector of length  $n$  is not easy, considering the fact that  $\mathbf{S}$  and  $\mathbf{R}$  do not know the identity of corrupted wires. Also it is very difficult for  $\mathbf{R}$  ( $\mathbf{S}$ ) to reliably send any information to  $\mathbf{S}$  ( $\mathbf{R}$ ). In the case of undirected graphs, there exists at least  $2t_b + 1$  bi-directional wires between  $\mathbf{S}$  and  $\mathbf{R}$  (which are necessary and sufficient for the existence of any SRMT/SSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ ) and so it is very easy to do reliable communication by simply sending the information through all the wires. However here, in the worst case, we may have  $t_b + 1$  and  $t_b$  wires in top and bottom band respectively. In protocol 3-SSMT-Static-Byzantine-Directed-Exponential,  $\mathbf{S}$  and  $\mathbf{R}$  could easily establish the vector, as they tried all possible subsets of size  $t_b + 1$  and there exists one subset consisting of all honest wires. However, we cannot use the same approach here, as we want to keep the computation and communication complexity polynomial. So we require completely different techniques to reliably and securely establish the keys.

Recall that during **Phase I**,  $\mathbf{S}$  has sent an  $(n + 1)$ -tuple over each wire in the top band. These tuples were not used in the first method (during **Phase I**) to send  $m^{\mathbf{S}}$ . So  $\mathbf{R}$  now use these tuples in the second phase.  $\mathbf{R}$  does not know which wires in the top band correctly delivered the  $(n + 1)$ -tuples. In order to facilitate  $\mathbf{S}$  to find out which tuples were delivered properly,  $\mathbf{R}$  hashes each received  $(n + 1)$ -tuple with a random hash key and sends back the hash values and hash keys to  $\mathbf{S}$  through the entire bottom band. Since there exists at least one honest wire in the bottom band, it will correctly deliver the hash keys and hash values. In addition to this,  $\mathbf{R}$  also sends a random  $(n + u)$ -tuple over each wire. Furthermore, each  $(n + u)$ -tuple is hashed by  $u$  random hash keys, one corresponding to each wire in the bottom band.

Till now, this part of the computation and communication is some what similar to what is done during protocol 3-SSMT-Static-Byzantine-Directed-Exponential. However, there are certain additional steps which are incorporated here. First of all,  $\mathbf{S}$  and  $\mathbf{R}$  have exchanged tuples over all the wires in top band and bottom band, rather than considering  $(t_b + 1)$ -sized subsets of wires. In addition to this,  $\mathbf{R}$  sends the hash value of each tuple received over the top band and each tuple sent over the bottom band. Finally, the size of the tuples which are exchanged over the top and bottom band are different. They are so to maintain the secrecy of  $m^{\mathbf{S}}$ , which will be delivered during third phase. More specifically, each tuple received over the top band is hashed by only

one key. So keeping the length of the tuples which are exchanged over top band as  $n + 1$  maintains the secrecy of its first  $n$  elements. On the other hand, each tuple in the bottom band is hashed by  $u$  random keys. So keeping the length of the tuples which are exchanged over bottom band as  $n + u$  maintains the secrecy of its first  $n$  elements.

Now as mentioned above, the information sent by  $\mathbf{R}$  may not reach reliably to  $\mathbf{S}$ . In the worst case, there can be only one honest wire in the bottom band but  $\mathbf{S}$  will not know its identity. Now according to the information received from  $\mathbf{R}$ , sender  $\mathbf{S}$  divides the bottom band into acceptable sets  $\mathcal{B}_1, \dots, \mathcal{B}_k$ , where  $1 \leq k \leq u$ . *The division is at the heart of the protocol.* Informally, the division is done as follows:  $\mathbf{S}$  considers a wire  $b_i$  in the bottom band and adds wire  $b_j$  from the bottom band in the set  $\mathcal{B}_i$  if  $b_j$  and  $b_i$  are *pairwise consistent*. Here, by pairwise consistency we mean that the hash key and hash value received by  $\mathbf{S}$  over wire  $b_j$  is consistent with the  $(n + u)$ -tuple received by  $\mathbf{S}$  over wire  $b_i$  and vice-versa. Notice that if  $b_j$  is an honest wire and if  $b_i$  and  $b_j$  are pairwise consistent, then with very high probability, the  $(n + u)$ -tuple received by  $\mathbf{S}$  over wire  $b_i$  is not modified. This is because the adversary does not know the hash key and hash value sent by  $\mathbf{R}$  over wire  $b_j$ , corresponding to the original  $i^{\text{th}}$   $(n + u)$ -tuple.

Now after computing the set  $\mathcal{B}_i$ ,  $\mathbf{S}$  computes the corresponding set  $\mathcal{F}_i$ . Recall that  $\mathbf{R}$  has sent the hash value of each  $(n + 1)$ -tuple received over the top band, through the entire bottom band. So  $\mathbf{S}$  considers the hash values (corresponding to the top band tuples), received over wire  $b_i$  and adds all such  $f_j$ 's in  $\mathcal{F}_i$ , such that the  $(n + 1)$ -tuple sent over wire  $f_j$  during **Phase I** is consistent with the  $j^{\text{th}}$  hash value, received over wire  $b_i$ . Notice that if  $f_j$  is a corrupted wire and if the  $(n + 1)$ -tuple sent over wire  $f_j$  is modified during **Phase I**, then with very high probability,  $f_j$  will not be added to  $\mathcal{F}_i$ , provided  $b_i$  is an honest wire. This is because if  $b_i$  is an honest wire then adversary will not know the  $j^{\text{th}}$  hash key and hash value, which is sent by  $\mathbf{R}$  over wire  $b_i$ .

Finally,  $\mathbf{S}$  considers the set  $\mathcal{B}_i$  as acceptable, if there are at least  $t_b + 1$  wires in total in  $\mathcal{F}_i$  and  $\mathcal{B}_i$ . It is easy to see that there will be at most  $u$  acceptable sets, one corresponding to each wire in the bottom band. Moreover, if  $b_i$  is an honest wire in the bottom band, then  $\mathcal{B}_i$  will always be an acceptable set, as all honest wires in top band and bottom band will be present in  $\mathcal{F}_i$  and  $\mathcal{B}_i$  respectively. It is this division of the bottom band, which makes  $\mathbf{S}$  now to work with  $u$  path sets, instead of  $\binom{2t_b+1}{t_b+1}$  path sets. The formal details of **Phase II** of protocol 3-SSMT-Static-Byzantine-Directed are given in Fig. 15.4.

**Remark 15.11** *The three phase efficient SSMT protocol of [24] as well as [54] also divides the bottom band into subsets during second phase, according to some what different criteria. In [24], as well as in [54], the authors claimed that their criteria will create at most  $u$  subsets of the bottom band. However, in the subsequent section, we will show that it is not so. In the worst case, there can be  $\mathcal{O}(3^u)$  subsets of the bottom band, thus making the communication and computational complexity of their protocol exponential. On the other hand, our criteria for division of bottom band always result in at most  $u$  subsets of the bottom band. It is this difference in the criteria of the division of the bottom band, which makes the communication and computational complexity of our protocol 3-SSMT-Static-Byzantine-Directed polynomial.*

We now prove the properties of **Phase II** of 3-SSMT-Static-Byzantine-Directed.

**Claim 15.12** *If  $b_i$  is an honest wire in the bottom band and  $b_i \in \mathcal{B}_j$ , corresponding to some wire  $b_j$  in the bottom band, then except with error probability  $2^{-\Omega(\kappa)}$ , the random  $(n + u)$ -tuple that  $\mathbf{S}$  has received along wire  $b_j$  is not modified.*

Figure 15.4: Phase II of Protocol 3-SSMT-Static-Byzantine-Directed

<p><b>Phase II: R to S:</b> R does the following computation and communication:</p> <ol style="list-style-type: none"> <li>1. For <math>i = 1, \dots, n</math>, R chooses a random, non-zero hash key <math>r_i^{\mathbf{R}}</math>, corresponding to wire <math>f_i</math>.</li> <li>2. R computes the set <math>\beta^{\mathbf{R}} = \{(r_i^{\mathbf{R}}, \gamma_i^{\mathbf{R}}) : i = 1, \dots, n\}</math>, where <math>\gamma_i^{\mathbf{R}} = \text{hash}(r_i^{\mathbf{R}}; x_{1,i}^{\mathbf{R}}, \dots, x_{n+1,i}^{\mathbf{R}})</math>. Recall that <math>(x_{1,i}^{\mathbf{R}}, \dots, x_{n+1,i}^{\mathbf{R}})</math> denotes the <math>(n+1)</math>-tuple which R has received during <b>Phase I</b> over wire <math>f_i</math>, for <math>i = 1, \dots, n</math>.</li> <li>3. For <math>i = 1, \dots, u</math>, R selects a random non-zero <math>(n+u)</math>-tuple <math>(y_{1,i}^{\mathbf{R}}, \dots, y_{n+u,i}^{\mathbf{R}})</math>.</li> <li>4. Corresponding to the <math>(n+u)</math>-tuple <math>(y_{1,i}^{\mathbf{R}}, \dots, y_{n+u,i}^{\mathbf{R}})</math>, R selects <math>u</math> random non-zero hash keys <math>\{\text{key}_{j,i}^{\mathbf{R}} : j = 1, \dots, u\}</math> from <math>\mathbb{F}</math>.</li> <li>5. For <math>i = 1, \dots, u</math>, R sends the following to S through wire <math>b_i</math>: <ol style="list-style-type: none"> <li>(a) <math>\beta^{\mathbf{R}}</math>;</li> <li>(b) The <math>(n+u)</math>-tuple <math>(y_{1,i}^{\mathbf{R}}, \dots, y_{n+u,i}^{\mathbf{R}})</math>;</li> <li>(c) The 2-tuple <math>(\text{key}_{j,i}^{\mathbf{R}}, \alpha_{j,i}^{\mathbf{R}})</math>, where <math>\alpha_{j,i}^{\mathbf{R}} = \text{hash}(\text{key}_{j,i}^{\mathbf{R}}; y_{1,i}^{\mathbf{R}}, \dots, y_{n+u,i}^{\mathbf{R}})</math>, for <math>j = 1, \dots, u</math>.</li> </ol> </li> </ol> <p><b>Computation by S at the end of Phase II:</b></p> <ol style="list-style-type: none"> <li>1. Let S receive the following over wire <math>b_i</math>, for <math>i = 1, \dots, u</math>: <ol style="list-style-type: none"> <li>(a) <math>\beta_i^{\mathbf{S}} = \{(r_{i,j}^{\mathbf{R}}, \gamma_{i,j}^{\mathbf{R}}) : j = 1, \dots, n\}</math>;</li> <li>(b) The <math>(n+u)</math>-tuple <math>(y_{1,i}^{\mathbf{S}}, \dots, y_{n+u,i}^{\mathbf{S}})</math>;</li> <li>(c) The 2-tuple <math>(\text{key}_{j,i}^{\mathbf{S}}, \alpha_{j,i}^{\mathbf{S}})</math>, for <math>j = 1, \dots, u</math>.</li> </ol> </li> <li>2. For <math>i = 1, \dots, u</math>, corresponding to wire <math>b_i</math>, S computes <math>\mathcal{B}_i</math> and <math>\mathcal{F}_i</math> as follows: <ol style="list-style-type: none"> <li>(a) S adds wire <math>b_j \in \{b_1, \dots, b_u\}</math> to <math>\mathcal{B}_i</math> (which is initially <math>\emptyset</math>) if <math>b_i, b_j</math> are found to be pair-wise consistent by satisfying both the following conditions: <ol style="list-style-type: none"> <li>i. <math>\alpha_{i,j}^{\mathbf{S}} = \text{hash}(\text{key}_{j,i}^{\mathbf{S}}; y_{1,i}^{\mathbf{S}}, \dots, y_{n+u,i}^{\mathbf{S}})</math>;</li> <li>ii. <math>\alpha_{j,i}^{\mathbf{S}} = \text{hash}(\text{key}_{j,i}^{\mathbf{S}}; y_{1,j}^{\mathbf{S}}, \dots, y_{n+u,j}^{\mathbf{S}})</math>.</li> </ol> </li> <li>(b) S adds wire <math>f_j</math> to <math>\mathcal{F}_i</math> (which is initially <math>\emptyset</math>), if <math>\gamma_{i,j}^{\mathbf{S}} = \text{hash}(r_{i,j}^{\mathbf{S}}; x_{1,j}^{\mathbf{S}}, \dots, x_{n+1,j}^{\mathbf{S}})</math>.</li> <li>(c) If <math> \mathcal{F}_i  +  \mathcal{B}_i  \leq t_b</math> then S concludes that <math>\mathcal{B}_i</math> is an unacceptable set, otherwise <math>\mathcal{B}_i</math> is an acceptable set.</li> </ol> </li> </ol>
--

PROOF: The claim holds without any error if wire  $b_j$  is honest. We now show that the claim even holds for a corrupted wire  $b_j$  with very high probability. So let  $b_j$  be a corrupted wire, such that the  $(n+u)$ -tuple received by S over wire  $b_j$  is modified; i.e.,  $(y_{1,j}^{\mathbf{S}}, \dots, y_{n+u,j}^{\mathbf{S}}) \neq (y_{1,j}^{\mathbf{R}}, \dots, y_{n+u,j}^{\mathbf{R}})$ . Since  $b_i \in \mathcal{B}_j$  is an honest wire, it implies that  $\text{key}_{j,i}^{\mathbf{S}} = \text{key}_{j,i}^{\mathbf{R}}$  and  $\alpha_{j,i}^{\mathbf{S}} = \alpha_{j,i}^{\mathbf{R}} = \text{hash}(\text{key}_{j,i}^{\mathbf{R}}; y_{1,i}^{\mathbf{R}}, \dots, y_{n+u,i}^{\mathbf{R}})$ . Moreover,  $\mathcal{A}_{t_b}^{\text{static}}$  will have no information about  $(\text{key}_{j,i}^{\mathbf{S}}, \alpha_{j,i}^{\mathbf{S}})$ . Furthermore, since  $b_i \in \mathcal{B}_j$ , it implies that  $\alpha_{j,i}^{\mathbf{S}} = \text{hash}(\text{key}_{j,i}^{\mathbf{S}}; y_{1,i}^{\mathbf{S}}, \dots, y_{n+u,i}^{\mathbf{S}})$ . However, from the properties of hashing, without knowing  $\text{key}_{j,i}^{\mathbf{S}}$ , the adversary can ensure that  $\text{hash}(\text{key}_{j,i}^{\mathbf{S}}; y_{1,i}^{\mathbf{S}}, \dots, y_{n+u,i}^{\mathbf{S}}) = \text{hash}(\text{key}_{j,i}^{\mathbf{S}}; y_{1,i}^{\mathbf{R}}, \dots, y_{n+u,i}^{\mathbf{R}})$ , even if  $(y_{1,i}^{\mathbf{S}}, \dots, y_{n+u,i}^{\mathbf{S}}) \neq (y_{1,i}^{\mathbf{R}}, \dots, y_{n+u,i}^{\mathbf{R}})$  with probability at most  $\frac{n+u}{|\mathbb{F}|-1} \approx 2^{-\Omega(\kappa)}$ , which is negligible in our context.  $\square$

**Claim 15.13** *If  $b_i$  is an honest wire in the bottom band then  $\mathcal{B}_i$  will be always considered as an acceptable set.*

PROOF: Let  $f_{i_1}, \dots, f_{i_T}$  and  $b_{i_1}, \dots, b_{i_B}$  denote the honest wires in the top and bottom band respectively. Now  $|\{f_{i_1}, \dots, f_{i_T}\}| + |\{b_{i_1}, \dots, b_{i_B}\}| \geq t_b + 1$ . Moreover  $|\{f_{i_1}, \dots, f_{i_T}\}| \geq 1$ . Furthermore according to the condition given in the claim,  $|\{b_{i_1}, \dots,$

$b_{i_B}\} \geq 1$ . To prove the claim we show that the wires  $b_{i_1}, \dots, b_{i_B}$  will be present in  $\mathcal{B}_i$  and the wires  $f_{i_1}, \dots, f_{i_T}$  will be present in the corresponding  $\mathcal{F}_i$ . So  $|\mathcal{F}_i| + |\mathcal{B}_i| = |\{f_{i_1}, \dots, f_{i_T}\}| + |\{b_{i_1}, \dots, b_{i_B}\}| \geq t_b + 1$  and thus  $\mathcal{B}_i$  will be an acceptable set.

First of all notice that all the wires  $b_{i_1}, \dots, b_{i_B}$  (including  $b_i$ ) will be present in  $\mathcal{B}_i$ . This is because any two wires  $b_j, b_k$  in the set  $\{b_{i_1}, \dots, b_{i_B}\}$  will be pairwise consistent because the following conditions are satisfied (as both these wires are honest):

1.  $\alpha_{j,k}^{\mathbf{S}} = \text{hash}(\text{key}_{j,k}^{\mathbf{S}}; y_{1,j}^{\mathbf{S}}, \dots, y_{n+u,j}^{\mathbf{S}})$ ;
2.  $\alpha_{k,j}^{\mathbf{S}} = \text{hash}(\text{key}_{k,j}^{\mathbf{S}}; y_{1,k}^{\mathbf{S}}, \dots, y_{n+u,k}^{\mathbf{S}})$ .

So the wires  $b_{i_1}, \dots, b_{i_B}$  will be present in  $\mathcal{B}_i$ . Since the wires  $f_{i_1}, \dots, f_{i_T}$  are honest, the  $(n+1)$ -tuple received by  $\mathbf{R}$  over these wires are the same as sent by  $\mathbf{S}$ . That is,  $(x_{1,j}^{\mathbf{R}}, \dots, x_{n+1,j}^{\mathbf{R}}) = (x_{1,j}^{\mathbf{S}}, \dots, x_{n+1,j}^{\mathbf{S}})$ , for every  $j \in \{i_1, \dots, i_T\}$ . This implies that  $\gamma_j^{\mathbf{R}} = \text{hash}(r_j^{\mathbf{R}}; x_{1,j}^{\mathbf{R}}, \dots, x_{n+1,j}^{\mathbf{R}}) = \text{hash}(r_j^{\mathbf{R}}; x_{1,j}^{\mathbf{S}}, \dots, x_{n+1,j}^{\mathbf{S}})$ , for every  $j \in \{i_1, \dots, i_T\}$ . Since the wires  $b_{i_1}, \dots, b_{i_B}$  are honest, they will correctly deliver  $\beta^{\mathbf{R}}$  and hence  $\beta_i^{\mathbf{S}} = \beta^{\mathbf{R}}$ , as wire  $b_i$  is honest. This implies that  $(\gamma_{i,j}^{\mathbf{S}}, r_{i,j}^{\mathbf{S}}) = (\gamma_j^{\mathbf{R}}, r_j^{\mathbf{R}})$  for every  $j \in \{i_1, \dots, i_T\}$ . So when  $\mathbf{S}$  executes step 2(b) of the computation at the end of **Phase II** with respect to  $\beta_i^{\mathbf{S}}$ , all the wires  $f_{i_1}, \dots, f_{i_T}$  will be added in  $\mathcal{F}_i$ .  $\square$

**Claim 15.14** *Let  $b_i$  be an honest wire in the bottom band. Then with very high probability, the  $(n+1)$ -tuple received by  $\mathbf{R}$  at the end of **Phase I** over the wires in  $\mathcal{F}_i$  are not modified.*

PROOF: Let  $b_i$  be an honest wire in the bottom band. Then  $|\mathcal{F}_i| \geq 1$ . This is because from the proof of the previous claim,  $|\mathcal{B}_i| + |\mathcal{F}_i| \geq t_b + 1$  and there can be at most  $t_b$  wires in  $\mathcal{B}_i$ . Now let  $f_j$  be an honest wire from the top band, which is present in  $\mathcal{F}_i$ . Since  $f_j$  is honest, it implies that it will correctly deliver the  $(n+1)$ -tuple to  $\mathbf{R}$ . On the other hand, let  $f_j$  be a corrupted wire in the top band, such that  $f_j$  has modified the  $(n+1)$ -tuple sent by  $\mathbf{S}$  over  $f_j$ . That is  $(x_{1,j}^{\mathbf{R}}, \dots, x_{n+1,j}^{\mathbf{R}}) \neq (x_{1,j}^{\mathbf{S}}, \dots, x_{n+1,j}^{\mathbf{S}})$ . We now show that except with probability  $2^{-\Omega(\kappa)}$ ,  $f_j$  will not be present in  $\mathcal{F}_i$ .

Notice that when modifying the  $(n+1)$ -tuple over wire  $f_j$ , the adversary has no idea about the random hash key  $r_j^{\mathbf{R}}$ , corresponding to wire  $f_j$ , which is going to be selected by  $\mathbf{R}$  during **Phase II**. Now  $\gamma_j^{\mathbf{R}} = \text{hash}(r_j^{\mathbf{R}}; x_{1,j}^{\mathbf{R}}, \dots, x_{n+1,j}^{\mathbf{R}})$ . Since wire  $b_i$  is honest, it will correctly deliver  $\beta^{\mathbf{R}}$  and hence  $(r_j^{\mathbf{R}}, \gamma_j^{\mathbf{R}})$ . So  $\beta_i^{\mathbf{S}} = \beta^{\mathbf{R}}$  and hence  $(r_{i,j}^{\mathbf{S}}, \gamma_{i,j}^{\mathbf{S}}) = (r_j^{\mathbf{R}}, \gamma_j^{\mathbf{R}})$ . Since  $(x_{1,j}^{\mathbf{R}}, \dots, x_{n+1,j}^{\mathbf{R}}) \neq (x_{1,j}^{\mathbf{S}}, \dots, x_{n+1,j}^{\mathbf{S}})$  and adversary has no information about  $r_j^{\mathbf{R}}$ , from the properties of hashing, except with probability  $\frac{n}{|\mathbb{F}|} \approx 2^{-\Omega(\kappa)}$ ,  $\gamma_{i,j}^{\mathbf{S}} \neq \text{hash}(r_{i,j}^{\mathbf{S}}; x_{1,j}^{\mathbf{S}}, \dots, x_{n+1,j}^{\mathbf{S}})$ . Thus except with probability  $2^{-\Omega(\kappa)}$ ,  $f_j$  will not be present in  $\mathcal{F}_i$ .  $\square$

**Claim 15.15** *Let  $\mathcal{B}_i$  be an acceptable set. If  $b_j$  is an honest wire in  $\mathcal{B}_i$ , then the adversary will have no information about the first  $n$  values from the  $(n+u)$ -tuple which is sent by  $\mathbf{R}$  over wire  $b_j$ . Similarly, if  $f_j$  is an honest wire in  $\mathcal{F}_i$ , then the adversary will have no information about the first  $n$  values from the  $(n+1)$ -tuple which is sent by  $\mathbf{S}$  over wire  $f_j$ .*

PROOF: Let  $\mathcal{B}_i$  be an acceptable set and let  $b_j$  be an honest wire in  $\mathcal{B}_i$ . Since  $b_j$  is an honest wire, the adversary will not know the  $(n+u)$ -tuple which  $\mathbf{R}$  will send over wire  $b_j$ . Now notice that during **Phase II**,  $\mathbf{R}$  hashes the  $(n+u)$ -tuple which is going to be sent over wire  $b_j$ , by  $u$  random hash keys and sends one (hash key, hash value) pair through each wire in the bottom band. In the worst case, the entire bottom

band, except wire  $b_j$  may be under the control of the adversary. So in the worst case, adversary will know  $u - 1$  distinct hash values, corresponding to the  $(n + u)$ -tuple sent over wire  $b_j$ . So from the properties of hashing, the adversary will have no information about the first  $n$  values from the  $(n + u)$ -tuple which is sent by  $\mathbf{R}$  over wire  $b_j$ .

Now suppose there exists an honest wire, say  $f_j$ , in  $\mathcal{F}_i$ . So the adversary will not know the  $(n + 1)$ -tuple which  $\mathbf{S}$  has sent over wire  $f_j$ . Now notice that during **Phase II**,  $\mathbf{R}$  hashes the  $(n + 1)$ -tuple received over wire  $f_j$  by a random hash key  $r_j^{\mathbf{R}}$ , adds the pair  $(r_j^{\mathbf{R}}, \gamma_j^{\mathbf{R}})$  to  $\beta^{\mathbf{R}}$  and sends  $\beta^{\mathbf{R}}$  over all the wires in the bottom band. So even if the entire bottom band is under the control of the adversary, the adversary will know *only one* hash value corresponding to the  $(n + 1)$ -tuple which was sent by  $\mathbf{S}$  over wire  $f_j$ . So from the properties of hashing, the adversary will have no information about the first  $n$  values from the  $(n + 1)$ -tuple which is sent by  $\mathbf{S}$  over wire  $f_j$ .  $\square$

We now summarize the properties of set  $\mathcal{B}_i, \mathcal{F}_i$ , corresponding to an honest wire  $b_i$  in the bottom band.

**Lemma 15.16** *Let  $b_i$  be an honest wire in the bottom band. Then  $\mathcal{B}_i$  and corresponding  $\mathcal{F}_i$  will have the following properties:*

1.  $\mathcal{B}_i$  will be an acceptable set.
2. All honest wires in the bottom band will be present in  $\mathcal{B}_i$ , while all the honest wires in the top band will be present in  $\mathcal{F}_i$ .
3. With very high probability, the  $(n + u)$ -tuple received by  $\mathbf{S}$  at the end of **Phase II** over the wires in  $\mathcal{B}_i$  are not modified.
4. With very high probability, the  $(n + 1)$ -tuple received by  $\mathbf{R}$  at the end of **Phase I** over the wires in  $\mathcal{F}_i$  are not modified.
5. The adversary will have no information about the first  $n$  values from the  $(n + u)$ -tuples which are exchanged over the honest wire(s) in  $\mathcal{B}_i$ . The adversary will also have no information about the first  $n$  values from the  $(n + 1)$ -tuples which are exchanged over the honest wire(s) in  $\mathcal{F}_i$ .

PROOF: Follows from the proof of Claim 15.12, Claim 15.13, Claim 15.14 and Claim 15.15.  $\square$

We now prove the properties of set  $\mathcal{B}_i, \mathcal{F}_i$ , corresponding to a wire  $b_i$  in the bottom band, such that  $b_i$  is under the control of the adversary.

**Claim 15.17** *Let  $b_i$  be a corrupted wire in the bottom band, such that there exists an honest wire in  $\mathcal{B}_i$ . Then the adversary can modify the  $(n + u)$ -tuple which are exchanged over the corrupted wires in  $\mathcal{B}_i$ , other than  $b_i$ , without letting  $\mathbf{S}$  know about it.*

PROOF: Let  $b_j$  be an honest wire present in  $\mathcal{B}_i$ . Then from the proof of Claim 15.12, the  $(n + u)$ -tuple is correctly exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  over wire  $b_i$  with very high probability. However, it does not imply that the  $(n + u)$ -tuple are correctly exchanged over other corrupted wires (if any) in  $\mathcal{B}_i$ . More specifically, let  $b_k$  be a corrupted wire in the bottom band, other than  $b_i$ , such that  $(y_{1,k}^{\mathbf{S}}, \dots, y_{n+u,k}^{\mathbf{S}}) \neq (y_{1,k}^{\mathbf{R}}, \dots, y_{n+u,k}^{\mathbf{R}})$ . Since  $b_i$  is also under the control of the adversary, the adversary will know  $(key_{k,i}^{\mathbf{R}}, \alpha_{k,i}^{\mathbf{R}})$ . Moreover, the adversary can modify the pair such that  $(key_{k,i}^{\mathbf{S}}, \alpha_{k,i}^{\mathbf{S}}) \neq (key_{k,i}^{\mathbf{R}}, \alpha_{k,i}^{\mathbf{R}})$  and  $\alpha_{k,i}^{\mathbf{S}} =$

$\text{hash}(\text{key}_{k,i}^{\mathbf{S}}; y_{1,k}^{\mathbf{S}}, \dots, y_{n+u,k}^{\mathbf{S}})$ . Furthermore, adversary does not modify  $(\text{key}_{i,k}^{\mathbf{R}}, \alpha_{i,k}^{\mathbf{R}})$  and thus  $(\text{key}_{i,k}^{\mathbf{S}}, \alpha_{i,k}^{\mathbf{S}}) = (\text{key}_{i,k}^{\mathbf{R}}, \alpha_{i,k}^{\mathbf{R}})$ , where  $\alpha_{i,k}^{\mathbf{S}} = \text{hash}(\text{key}_{i,k}^{\mathbf{S}}; y_{1,i}^{\mathbf{S}}, \dots, y_{n+u,i}^{\mathbf{S}})$ . If the adversary behaves in this manner, then  $b_i$  and  $b_k$  will be pairwise consistent, while  $b_k$  and  $b_j$  will not be pair-wise consistent. But still  $b_k$  will be included in  $\mathcal{B}_i$  and neither  $\mathbf{S}$  nor  $\mathbf{R}$  will know that the  $(n+u)$ -tuple exchanged over wire  $b_k \in \mathcal{B}_i$  is corrupted.  $\square$

**Claim 15.18** *Let  $b_i$  be a corrupted wire in the bottom band and let  $f_j$  be a corrupted wire in the top band, which is present in  $\mathcal{F}_i$ . Then the adversary can modify the  $(n+1)$ -tuple which is exchanged over corrupted wire  $f_j$ , without letting  $\mathbf{S}$  and  $\mathbf{R}$  know about it.*

PROOF: Let  $f_j$  be a corrupted wire in the top band, such that  $(x_{1,j}^{\mathbf{R}}, \dots, x_{n+1,j}^{\mathbf{R}}) \neq (x_{1,j}^{\mathbf{S}}, \dots, x_{n+1,j}^{\mathbf{S}})$ . Since  $b_i$  is also under the control of the adversary, it implies that  $\beta^{\mathbf{R}}$  and hence  $(r_j^{\mathbf{R}}, \gamma_j^{\mathbf{R}})$  is also known to the adversary. Now notice that  $\gamma_j^{\mathbf{R}} = \text{hash}(r_j^{\mathbf{R}}; x_{1,j}^{\mathbf{R}}, \dots, x_{n+1,j}^{\mathbf{R}})$ . During the transmission of  $\beta^{\mathbf{R}}$  over  $b_i$ , the adversary can simply change the  $j^{\text{th}}$  pair in  $\beta^{\mathbf{R}}$ , such that  $(r_{i,j}^{\mathbf{S}}, \gamma_{i,j}^{\mathbf{S}}) \neq (r_j^{\mathbf{R}}, \gamma_j^{\mathbf{R}})$  and  $\gamma_{i,j}^{\mathbf{S}} = \text{hash}(r_{i,j}^{\mathbf{S}}; x_{1,j}^{\mathbf{S}}, \dots, x_{n+1,j}^{\mathbf{S}})$ . The adversary can do so because he knows  $(x_{1,j}^{\mathbf{S}}, \dots, x_{n+1,j}^{\mathbf{S}})$  and  $r_j^{\mathbf{R}}$  and  $b_i$  is under his control. So even though the  $(n+1)$ -tuple is not correctly exchanged over  $f_j$ , still  $f_j$  can be present in  $\mathcal{B}_i$ .  $\square$

We now summarize the properties of set  $\mathcal{B}_i, \mathcal{F}_i$ , corresponding to a wire  $b_i$  in the bottom band, such that  $b_i$  is under the control of the adversary.

**Lemma 15.19** *Let  $b_i$  be a wire in the bottom band such that  $b_i$  is under the control of the adversary. Moreover, let  $\mathcal{B}_i$  be an acceptable set. Then  $\mathcal{B}_i$  and corresponding  $\mathcal{F}_i$  will have the following properties:*

1. *There will exist at least one honest wire, either from the top band or bottom band, which will be present in  $\mathcal{F}_i$  or  $\mathcal{B}_i$  respectively.*
2. *If there exists an honest wire in  $\mathcal{F}_i$ , then the  $(n+1)$ -tuple is correctly exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  over that wire. Moreover, adversary will have no information about the first  $n$  values of the  $(n+1)$ -tuple.*
3. *If there exists an honest wire in  $\mathcal{B}_i$ , then the  $(n+u)$ -tuple is correctly exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  over that wire. Moreover, adversary will have no information about the first  $n$  values of the  $(n+u)$ -tuple.*
4. *If there exists an honest wire in  $\mathcal{B}_i$ , then with very high probability the  $(n+u)$ -tuple is correctly exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  over wire  $b_i$ . However, the adversary can modify the  $(n+u)$ -tuple which are exchanged over the corrupted wires in  $\mathcal{B}_i$ , other than  $b_i$ , without letting  $\mathbf{S}$  know about it.*
5. *If there is no honest wire in  $\mathcal{B}_i$ , then the adversary can always modify the  $(n+u)$ -tuple which are exchanged over the corrupted wires in  $\mathcal{B}_i$ , without letting  $\mathbf{S}$  know about it.*
6. *Irrespective of the number of honest wires in the top band, the adversary can always modify the  $(n+1)$ -tuple which is exchanged over a corrupted wire (if any) in  $\mathcal{F}_i$ , without letting  $\mathbf{S}$  and  $\mathbf{R}$  know about it.*

PROOF: Since  $\mathcal{B}_i$  is an acceptable set, it implies that  $|\mathcal{B}_i| + |\mathcal{F}_i| \geq t_b + 1$ . In the worst case there can  $t_b$  corrupted wires including top and bottom band. This implies that there exists at least one honest wire, which is present either in  $\mathcal{F}_i$  or  $\mathcal{B}_i$ . This proves the first property.

Let  $f_j$  be an honest wire present in  $\mathcal{F}_i$ . It implies that the  $(n+1)$ -tuple is correctly exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  over wire  $f_j$ . Moreover, from the proof of Claim 15.15, adversary will have no information about the first  $n$  values of the  $(n+1)$ -tuple. This proves the second property.

Let  $b_j$  be an honest wire present in  $\mathcal{B}_i$ . It implies that the  $(n+u)$ -tuple is correctly exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  over wire  $b_j$ . Moreover, from the proof of Claim 15.15, adversary will have no information about the first  $n$  values of the  $(n+u)$ -tuple. This proves the third property.

The fourth property follows from the proof of Claim 15.17.

We now prove the fifth property. The corrupted wires in the bottom band can behave in such a way that even though there does not exist any honest wire in  $\mathcal{B}_i$ , still  $\mathcal{B}_i$  becomes an acceptable set. We consider a possible setting and adversarial behavior in which it is possible. The setting can be easily generalized. More specifically, suppose  $u = t_b$  and there exists  $t_b - 1$  corrupted wires in the bottom band, who modify the  $(n+u)$ -tuple exchanged over them. Moreover, as explained in the proof of Claim 15.17, the adversary can control these  $t_b - 1$  wires in such a way that from  $\mathbf{S}$ 's point of view, all the  $t_b - 1$  corrupted wires are pairwise consistent. Furthermore, adversary can ensure that none of these  $t_b - 1$  corrupted wires are pair wise consistent with the single honest wire which is present in the bottom band. That is, if  $b_j$  is an honest wire in the bottom band, then the adversary can simply modify  $(key_{j,k}^{\mathbf{R}}, \alpha_{j,k}^{\mathbf{R}})$ , over all  $t_b - 1$  corrupted  $b_k$ 's, thus making  $b_j$  not pair-wise consistent with any of the  $t_b - 1$  corrupted wires. Now as a result of such adversarial behavior, only corrupted wires (which are  $t_b - 1$ ) will be present in  $\mathcal{B}_i$ . In order that  $\mathcal{B}_i$  becomes acceptable, there should be at least two wires in  $\mathcal{F}_i$ . Notice that in the scenario we are considering, there are  $t_b$  honest wires in the top band, who will correctly deliver the  $(n+1)$ -tuples to  $\mathbf{R}$ . So if  $b_i$  correctly delivers  $\beta_i^{\mathbf{R}}$  without doing any modification, then  $\beta_i^{\mathbf{S}} = \beta_i^{\mathbf{R}}$  and hence  $\mathbf{S}$  will include all the  $t_b$  honest wires from the top band in  $\mathcal{F}_i$  and hence  $\mathcal{B}_i$  will become acceptable. This proves the fifth property.

The last property follows from the proof of Claim 15.18.  $\square$

**Remark 15.20 (Difference Between Corrupted and Honest Acceptable Set)**

*Comparing Lemma 15.16 and Lemma 15.19, we find that in case of honest  $b_i$ , the  $(n+1)$ -tuples and  $(n+u)$ -tuples are exchanged correctly between  $\mathbf{S}$  and  $\mathbf{R}$  over all the wires in  $\mathcal{F}_i$  and  $\mathcal{B}_i$  respectively with very high probability. Moreover, there will be at least  $t_b + 1$  (honest) wires distributed in  $\mathcal{F}_i$  and  $\mathcal{B}_i$ , such that adversary will have no information about the first  $n$  values of the tuples that are exchanged over those wires. On the other hand, in the case of corrupted  $b_i$ , the  $(n+1)$ -tuples and  $(n+u)$ -tuples are exchanged correctly between  $\mathbf{S}$  and  $\mathbf{R}$  only over the honest wires in  $\mathcal{F}_i$  and  $\mathcal{B}_i$  respectively. Moreover, there will be at least one (honest) wire either in  $\mathcal{F}_i$  or  $\mathcal{B}_i$ , such that the adversary will have no information about the first  $n$  values of the tuple that is exchanged over that wire.*

**Lemma 15.21** *In protocol 3-SSMT-Static-Byzantine-Directed, there can be at most  $u$  acceptable sets.*

PROOF: The proof follows from the fact corresponding to each wire  $b_i$  in the bottom band, there is only one  $\mathcal{B}_i$  and  $\mathcal{F}_i$ .  $\square$

**Lemma 15.22** *In protocol 3-SSMT-Static-Byzantine-Directed,  $\mathbf{R}$  communicates  $\mathcal{O}(n^2\kappa)$  bits during Phase II.*

PROOF: During **Phase II**,  $\mathbf{R}$  sends  $\mathcal{O}(n)$  field elements over each wire. This requires a communication complexity of  $\mathcal{O}(nu\kappa) = \mathcal{O}(n^2\kappa)$  bits, as  $u = \mathcal{O}(n)$ .  $\square$

Finally before proceeding further to the description of third phase of the protocol, we note that  $\mathbf{R}$  executes **Phase II** only if he could not recover  $m^{\mathbf{S}}$  at the end of **Phase I**. However, even if  $\mathbf{R}$  recovers  $m^{\mathbf{S}}$  at the end of **Phase I**, he has no way to signal this to  $\mathbf{S}$ . This is because there are at most  $t_b$  wires in the bottom band and in the worst case, entire bottom band may be under the control of the adversary. So even if  $\mathbf{R}$  does not execute **Phase II** and does no communication to  $\mathbf{S}$ , in the worst case the adversary may simply control some wire(s) in the bottom band and pass some information to  $\mathbf{S}$ , which may lead  $\mathbf{S}$  to construct some valid acceptable set(s), corresponding to those wire(s)!! Fortunately the properties of hashing ensures that this happens with very negligible probability, as shown in the next lemma.

**Lemma 15.23** *Suppose in protocol 3-SSMT-Static-Byzantine-Directed,  $\mathbf{R}$  recovers  $m^{\mathbf{S}}$  at the end of **Phase I** and does no communication over the bottom band. Then the probability that the adversary corrupts bottom band and sends some arbitrary information to  $\mathbf{S}$  which leads to the construction of acceptable set is at most  $2^{-\Omega(\kappa)}$ .*

PROOF: Consider the following settings: there are  $t_b + 2$  wires in the top band and  $t_b - 1$  wires in the bottom band, such that the entire bottom band and one wire from the top band are under the control of the adversary. Suppose  $\mathbf{R}$  recovers  $m^{\mathbf{S}}$  at the end of **Phase I** itself and so does not execute **Phase II**. However, the adversary may do the following: the adversary selects  $t_b - 1$  arbitrary  $(n + u)$ -tuples and  $(t_b - 1)^2$  random hash keys and put these values over the wires in the bottom band in such a way, as if  $\mathbf{R}$  has executed **Phase II** using the  $(n + u)$ -tuples and  $(t_b - 1)^2$  random hash keys. So it is easy to see that all the wires in the bottom band will be pairwise consistent and thus  $|\mathcal{B}_i| = t_b - 1$ , for  $i = 1, \dots, t_b - 1$ . However, in order that any of these  $\mathcal{B}_i$  becomes an acceptable set, the adversary has to ensure that the corresponding  $|\mathcal{F}_i| \geq 2$ .

Now notice that the adversary will know the  $(n + 1)$ -tuple which is exchanged over the single wire in the top band which is under the control of the adversary. So the adversary can *always* produce the hash value of this tuple, corresponding to any hash key. However, the adversary will have no information about the  $(n + 1)$ -tuples which are exchanged over the  $t_b + 1$  honest wires in the top band. The probability that the adversary will be able to produce the hash value of the  $(n + 1)$ -tuple, corresponding to any of these  $t_b + 1$  honest wires, for a given hash key is same as the probability of correctly guessing the corresponding  $(n + 1)$ -tuple, which is  $\frac{1}{|\mathbb{F}|^{n+1}} \approx 2^{-\Omega(\kappa)}$ .

Now the adversary may do the following: he selects  $t_b + 2$  hash keys, one corresponding to each wire in the top band. The adversary also guesses the  $(n + 1)$ -tuple which  $\mathbf{S}$  would have sent over each honest wire in the top band and computes the hash value of those tuples, as if the tuples are received by  $\mathbf{R}$ . The adversary also computes the hash value of the  $(n + 1)$ -tuple, which  $\mathbf{S}$  has sent over the wire under its control in the top band. Thus the adversary computes  $\beta^{\mathbf{R}}$ , as if  $\beta^{\mathbf{R}}$  is computed by  $\mathbf{R}$ . The adversary is sure that at least one (hash-key, hash-value) pair in the computed  $\beta^{\mathbf{R}}$ , namely the one corresponding to the corrupted wire in the top band is correct. The remaining (hash-key, hash-value) pair in the computed  $\beta^{\mathbf{R}}$  (corresponding to the honest wires in the top band) may be correct, depending upon the guess of the adversary. The adversary then sends the computed  $\beta^{\mathbf{R}}$  to  $\mathbf{S}$  over the entire bottom band. On receiving

$\beta_i^{\mathbf{S}} = \beta^{\mathbf{R}}$ ,  $\mathbf{S}$  will add the corrupted wire in the top band to the set  $\mathcal{F}_i$ . Moreover, if the adversary has successfully guessed the  $(n + 1)$ -tuple which was sent over some honest wire  $f_j$ , then the  $j^{\text{th}}$  (hash-key, hash-value) pair in  $\beta_i^{\mathbf{S}}$  would be correct and hence  $f_j$  will be added in  $\mathcal{F}_i$ , thus making  $|\mathcal{F}_i| = 2$  and hence  $\mathcal{B}_i$  as an acceptable set. However, this happens with probability  $2^{-\Omega(\kappa)}$ .  $\square$

We now proceed towards the discussion of third phase, which is given in the next subsection.

### 15.5.2.3 Phase III of Protocol 3-SSMT-Static-Byzantine-Directed

Notice that at the end of second phase,  $\mathbf{S}$  could have  $u$  acceptable sets.  $\mathbf{S}$  will not know which of these acceptable sets contains only honest wires. From  $\mathbf{S}$ 's view point, all acceptable set may look valid. So  $\mathbf{S}$  assumes that all the acceptable sets are valid and tries to compute separate encryption key and authentication key from each acceptable set. Specifically,  $\mathbf{S}$  considers the wires in each acceptable  $\mathcal{B}_i$  and corresponding  $\mathcal{F}_i$  as a valid path set and using them,  $\mathbf{S}$  does the same computation and communication as done in **Phase III** of protocol 3-SSMT-Static-Byzantine-Directed-Exponential. However, instead of dealing with  $\binom{2t_b+1}{t_b+1}$  path sets,  $\mathbf{S}$  has to only consider  $u$  path sets. In the same way,  $\mathbf{R}$  recovers the message at the end of third phase by performing similar computation, as it does at the end of **Phase III** of 3-SSMT-Static-Byzantine-Directed-Exponential.

The secrecy of the protocol follows from the fact that corresponding to every acceptable  $\mathcal{B}_i$ , there exists at least one honest wire, either in  $\mathcal{B}_i$  or corresponding  $\mathcal{F}_i$ , such that the adversary will have no information about the first  $n$  values of the tuple which is exchanged correctly between  $\mathbf{S}$  and  $\mathbf{R}$  over the honest wire. So adversary will have no information about the  $n$ -tuple (whose elements are considered as the authentication and encryption keys), computed by  $\mathbf{S}$  from the tuples, which are exchanged over the wires in  $\mathcal{B}_i$  and  $\mathcal{F}_i$ . Moreover, if the tuples are not correctly exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  over the wires in  $\mathcal{B}_i$  and  $\mathcal{F}_i$ , then  $\mathbf{S}$  and  $\mathbf{R}$  will end up with different version of authentication and encryption keys. So except with negligible error probability, the verification at  $\mathbf{R}$ 's end will fail. This ensures reliability. The **Phase III** of protocol 3-SSMT-Static-Byzantine-Directed is formally presented in Fig. 15.5.

We now prove the properties of **Phase III** of protocol 3-SSMT-Static-Byzantine-Directed.

**Lemma 15.24** *In protocol 3-SSMT-Static-Byzantine-Directed,  $m^{\mathbf{S}}$  will be information theoretically secure at the end of **Phase III**.*

**PROOF:** In protocol 3-SSMT-Static-Byzantine-Directed, during **Phase III**,  $\mathbf{S}$  encrypts and authenticates  $m^{\mathbf{S}}$  by using the tuples, which are exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  over the wires in  $\mathcal{B}_l$  and  $\mathcal{F}_l$ , if  $\mathcal{B}_l$  is an acceptable set. Now as stated in Remark 15.20, irrespective of whether wire  $b_l$  is honest or corrupted, there exists at least one honest wire, either in  $\mathcal{B}_l$  or  $\mathcal{F}_l$ , such that the adversary will have no information about the first  $n$  elements of the tuple which is exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  over that honest wire. Since the keys  $\mathcal{C}_{1,l}^{\mathbf{S}}, \dots, \mathcal{C}_{n,l}^{\mathbf{S}}$  are computed by adding the first  $n$  elements of the all the tuples which are exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  over the wires in  $\mathcal{B}_l$  or  $\mathcal{F}_l$ , it implies that  $\mathcal{C}_{1,l}^{\mathbf{S}}, \dots, \mathcal{C}_{n,l}^{\mathbf{S}}$  will be information theoretically secure. Since  $\mathcal{C}_{1,l}^{\mathbf{S}}, \dots, \mathcal{C}_{n,l}^{\mathbf{S}}$  are used as encryption and authentication keys, by the properties of  $US_{auth}$ , it follows that  $m^{\mathbf{S}}$  will be information theoretically secure at the end of **Phase III**.  $\square$

Figure 15.5: Phase III of Protocol 3-SSMT-Static-Byzantine-Directed

**Phase III: S to R:** For each acceptable set  $\mathcal{B}_l$  and corresponding set  $\mathcal{F}_l$ ,  $\mathbf{S}$  does the following computation and communication:

1.  $\mathbf{S}$  considers the first  $n$  elements from the  $(n+1)$ -tuples which it had sent over the wires in  $\mathcal{F}_l$  during **Phase I** and the first  $n$  elements from the  $(n+u)$ -tuples which  $\mathbf{S}$  has received over the wires in  $\mathcal{B}_l$  during **Phase II**. By using them,  $\mathbf{S}$  computes his version of  $n$  keys  $\mathcal{C}_{1,l}^{\mathbf{S}} = \sum_{f_j \in \mathcal{F}_l} x_{1,j}^{\mathbf{S}} + \sum_{b_j \in \mathcal{B}_l} y_{1,j}^{\mathbf{S}}$ ,  $\mathcal{C}_{2,l}^{\mathbf{S}} = \sum_{f_j \in \mathcal{F}_l} x_{2,j}^{\mathbf{S}} + \sum_{b_j \in \mathcal{B}_l} y_{2,j}^{\mathbf{S}}$ ,  $\dots$ ,  $\mathcal{C}_{n,l}^{\mathbf{S}} = \sum_{f_j \in \mathcal{F}_l} x_{n,j}^{\mathbf{S}} + \sum_{b_j \in \mathcal{B}_l} y_{n,j}^{\mathbf{S}}$ .
2. For each element of  $m^{\mathbf{S}}$  (recall that  $|m^{\mathbf{S}}| = \frac{n}{3}$ ),  $\mathbf{S}$  takes three elements from the keys computed in the previous step and computes the set  $\mathcal{S}_l^{\mathbf{S}} = \{(c_{i,l}^{\mathbf{S}}, d_{i,l}^{\mathbf{S}}) : i = 1, \dots, \frac{n}{3}\}$  where  $(c_{i,l}^{\mathbf{S}}, d_{i,l}^{\mathbf{S}}) = USauth(m_i^{\mathbf{S}}; \mathcal{C}_{3i-2,l}^{\mathbf{S}}, \mathcal{C}_{3i-1,l}^{\mathbf{S}}, \mathcal{C}_{3i,l}^{\mathbf{S}})$ , for  $i = 1, \dots, \frac{n}{3}$ .
3.  $\mathbf{S}$  sends the set  $\mathcal{F}_l, \mathcal{B}_l$  and  $\mathcal{S}_l^{\mathbf{S}}$  to  $\mathbf{R}$  over all the wires in the set  $\mathcal{F}_l$  and terminates the protocol.

**Message Recovery by R:**

1. Let  $\mathbf{R}$  receive the sets  $\mathcal{F}_{j,l}^{\mathbf{R}}, \mathcal{B}_{j,l}^{\mathbf{R}}$  and  $\mathcal{S}_{j,l}^{\mathbf{R}}$  along wire  $f_j$ , for  $j = 1, \dots, n$ . In the worst case,  $l = 1, \dots, u$ .  $\mathbf{R}$  then executes the following steps.
2. If for some  $j \in \{1, 2, \dots, n\}$  and some  $l \in \{1, 2, \dots, u\}$ ,  $|\mathcal{F}_{j,l}^{\mathbf{R}}| + |\mathcal{B}_{j,l}^{\mathbf{R}}| \leq t_b$ , then  $\mathbf{R}$  concludes that wire  $f_j$  is corrupted and neglects wire  $f_j$ .
3. If  $f_j$  is not neglected, then for each  $\mathcal{F}_{j,l}^{\mathbf{R}}, \mathcal{B}_{j,l}^{\mathbf{R}}$  and  $\mathcal{S}_{j,l}^{\mathbf{R}}$  received along wire  $f_j$ ,  $\mathbf{R}$  does the following:
  - (a) Let  $\mathcal{S}_{j,l}^{\mathbf{R}} = \{(c_{j,i,l}^{\mathbf{R}}, d_{j,i,l}^{\mathbf{R}}) : i = 1, \dots, \frac{n}{3}\}$ .
  - (b) By using the index of the wires in  $\mathcal{F}_{j,l}^{\mathbf{R}}$  and  $\mathcal{B}_{j,l}^{\mathbf{R}}$ ,  $\mathbf{R}$  computes his version of  $n$  keys  $\mathcal{C}_{j,1,l}^{\mathbf{R}}, \dots, \mathcal{C}_{j,n,l}^{\mathbf{R}}$ .
  - (c) For  $i = 1, \dots, \frac{n}{3}$ ,  $\mathbf{R}$  applies the verification process of  $USauth$  on  $c_{j,i,l}^{\mathbf{R}}, d_{j,i,l}^{\mathbf{R}}, \mathcal{C}_{j,3i-2,l}^{\mathbf{R}}, \mathcal{C}_{j,3i-1,l}^{\mathbf{R}}$  and  $\mathcal{C}_{j,3i,l}^{\mathbf{R}}$ .
  - (d) If the verification is successful for all  $i = 1, \dots, \frac{n}{3}$ , then  $\mathbf{R}$  recovers  $m_{j,i,l}^{\mathbf{R}}$  from  $c_{j,i,l}^{\mathbf{R}}$ , for  $i = 1, \dots, \frac{n}{3}$ .
  - (e) Finally,  $\mathbf{R}$  concatenates  $m_{j,1,l}^{\mathbf{R}}, \dots, m_{j,\frac{n}{3},l}^{\mathbf{R}}$  to reconstruct the message  $m^{\mathbf{R}}$  and terminates the protocol.

**Lemma 15.25** *In protocol 3-SSMT-Static-Byzantine-Directed,  $m^{\mathbf{R}} = m^{\mathbf{S}}$ , except with error probability  $2^{-\Omega(\kappa)}$ .*

PROOF: Let  $\mathcal{F}_{j,l}^{\mathbf{R}}, \mathcal{B}_{j,l}^{\mathbf{R}}$  and  $\mathcal{S}_{j,l}^{\mathbf{R}}$  denote the sets, which passes the verification test in step 3 of message recovery. This implies that  $\mathbf{R}$  has recovered  $m^{\mathbf{R}}$  from  $\mathcal{S}_{j,l}^{\mathbf{R}}$  after computing his keys from the tuples which are exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  over the wires in  $\mathcal{F}_{j,l}^{\mathbf{R}}$  and  $\mathcal{B}_{j,l}^{\mathbf{R}}$ . Now notice that there exists at least one honest wire, which is present in either  $\mathcal{F}_{j,l}^{\mathbf{R}}$  or  $\mathcal{B}_{j,l}^{\mathbf{R}}$ , such that adversary will have no information about the first  $n$  elements of the tuple exchanged over that wire. So the keys computed by  $\mathbf{R}$  from

$\mathcal{F}_{j,l}^{\mathbf{R}}$  and  $\mathcal{B}_{j,l}^{\mathbf{R}}$  will be information theoretically secure. Moreover, as explained in the previous lemma, the keys which are used by  $\mathbf{S}$  for encrypting and authenticating  $m^{\mathbf{S}}$  will also be information theoretically secure. Now irrespective of whether the tuples are correctly exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  over all the wires in  $\mathcal{F}_{j,l}^{\mathbf{R}}$  and  $\mathcal{B}_{j,l}^{\mathbf{R}}$ , the adversary will have no information about the keys used by  $\mathbf{S}$  and the keys used by  $\mathbf{R}$ . So from the properties of *USauth*, if at all  $\mathbf{R}$  outputs  $m^{\mathbf{R}}$  from  $\mathcal{S}_{j,l}^{\mathbf{R}}$ , then except with probability  $2^{-\Omega(\kappa)}$ ,  $m^{\mathbf{R}} = m^{\mathbf{S}}$ .  $\square$

**Lemma 15.26** *In protocol 3-SSMT-Static-Byzantine-Directed, there always exist a wire  $f_j$  in the top band, such that  $\mathcal{F}_{j,l}^{\mathbf{R}}, \mathcal{B}_{j,l}^{\mathbf{R}}$  and  $\mathcal{S}_{j,l}^{\mathbf{R}}$  received by  $\mathbf{R}$  over wire  $f_j$  will satisfy the verification process.*

PROOF: The proof simply follows from the fact that there exists at least  $t_b + 1$  honest wires including top and bottom band and these  $t_b + 1$  honest wires will form some acceptable set  $\mathcal{B}_l$  and corresponding  $\mathcal{F}_l$ . The rest now follows from the protocol code for **Phase III**.  $\square$

**Lemma 15.27** *During Phase III,  $\mathbf{S}$  communicates  $\mathcal{O}(n^3\kappa)$  bits.*

PROOF: During **Phase III**, corresponding to an acceptable set  $\mathcal{B}_l$ ,  $\mathbf{S}$  sends the identity of  $\mathcal{B}_l, \mathcal{F}_l$  and the encryption, authentication of  $m^{\mathbf{S}}$  along all the wires in  $\mathcal{F}_l$ . This requires a communication complexity of  $\mathcal{O}(n^2)$  field elements and hence  $\mathcal{O}(n^2\kappa)$  bits. Now there can be  $u = \mathcal{O}(n)$  acceptable sets. So the worst case communication complexity of **Phase III** is  $\mathcal{O}(n^3\kappa)$  bits.  $\square$

We now summarize the properties of protocol 3-SSMT-Static-Byzantine-Directed by the following theorem.

**Theorem 15.28** *Protocol 3-SSMT-Static-Byzantine-Directed is a valid SSMT protocol, which sends a message containing  $\frac{n}{3}\kappa$  bits by communicating  $\mathcal{O}(n^3\kappa)$  bits and takes at most three phases.*

Finally, before ending our discussion on three phase SSMT, we discuss about the three phase SSMT protocol of [24] and [54] and show that the computational and communication complexity of both the protocols are exponential, as opposed to polynomial, as claimed in [24] and [54].

### 15.5.3 Inefficiency of the Three Phase SSMT Protocol of [24, 54, 87]

In [54], the authors presented a three phase SSMT protocol called  $\Pi_{modified}^{Existing}$  (see Page 314 of [54]). The protocol securely sends a message containing  $\frac{n}{3}$  field elements. The idea of the protocol is similar to the three phase SSMT protocol of [24, 87]<sup>4</sup> except that the SSMT protocol of [24, 87] sends a single message. The authors in [54] called the three phase SSMT protocol of [24, 87] as  $\Pi^{Existing}$ . The authors in [54], as well as in [24, 87] claimed that their three phase SSMT protocol requires polynomial computational and communication complexity<sup>5</sup>. However, we now show that the computational and communication complexity of the SSMT protocols of [24, 54, 87] are exponential.

<sup>4</sup>The three phase SSMT protocol presented in [24] and [87] are same.

<sup>5</sup>[54] claimed that their three phase SSMT protocol requires a communication complexity of  $\mathcal{O}(n^3\kappa)$  bits, where as [24, 87] claimed that their SSMT protocol is efficient, requiring polynomial computational and communication complexity.

We specifically consider protocol  $\Pi_{modified}^{Existing}$  and show an adversarial behavior, which may result  $\mathbf{S}$  to communicate exponential number of bits during third phase. The behavior also causes  $\mathbf{S}$  and  $\mathbf{R}$  to perform exponential computation during **Phase III** and at the end of **Phase III** respectively. A similar behavior can cause the same result for the SSMT protocol  $\Pi^{Existing}$ .

The three phase SSMT protocol  $\Pi_{modified}^{Existing}$  of [54] is same as protocol 3-SSMT-Static-Byzantine-Directed presented in the previous section, except for the computation which is done by  $\mathbf{S}$  at the end of **Phase II**. Specifically, in protocol  $\Pi_{modified}^{Existing}$ ,  $\mathbf{S}$  divides the bottom band at the end of **Phase II**, using some what different criteria, as shown in Fig. 15.6.

Figure 15.6: Computation by  $\mathbf{S}$  at the End of Phase II of Protocol  $\Pi_{modified}^{Existing}$  in [54]

**S does the Following Computation at the End of Phase II:**

1. Let  $\mathbf{S}$  receive the following over wire  $b_i$ , for  $i = 1, \dots, u$ :
  - (a)  $\beta_i^{\mathbf{S}} = \{(r_{i,j}^{\mathbf{R}}, \gamma_{i,j}^{\mathbf{R}}) : j = 1, \dots, n\}$ ;
  - (b) The  $(n + u)$ -tuple  $(y_{1,i}^{\mathbf{S}}, \dots, y_{n+u,i}^{\mathbf{S}})$ ;
  - (c) The 2-tuple  $(key_{j,i}^{\mathbf{S}}, \alpha_{j,i}^{\mathbf{S}})$ , for  $j = 1, \dots, u$ .
2.  $\mathbf{S}$  divides the bottom band  $\{b_1, \dots, b_u\}$  into subsets  $\mathcal{B}_1, \dots, \mathcal{B}_k$ , such that for  $l = 1, \dots, k$ , every two wires  $b_i, b_j \in \mathcal{B}_l$  are pair-wise consistent by satisfying the following conditions:
  - (a)  $\beta_i^{\mathbf{S}} = \beta_j^{\mathbf{S}}$ ;
  - (b)  $\alpha_{i,j}^{\mathbf{S}} = hash(key_{i,j}^{\mathbf{S}}; y_{1,i}^{\mathbf{S}}, \dots, y_{n+u,i}^{\mathbf{S}})$ ;
  - (c)  $\alpha_{j,i}^{\mathbf{S}} = hash(key_{j,i}^{\mathbf{S}}; y_{1,j}^{\mathbf{S}}, \dots, y_{n+u,j}^{\mathbf{S}})$ .
3. For  $l = 1, \dots, k$ ,  $\mathbf{S}$  computes the set  $\mathcal{F}_l$ , corresponding to  $\mathcal{B}_l$  as follows:
  - (a) Let  $b_i \in \mathcal{B}_l$ .
  - (b)  $\mathbf{S}$  adds wire  $f_j$  in  $\mathcal{F}_l$  if  $\gamma_{i,j}^{\mathbf{S}} = hash(r_{i,j}^{\mathbf{S}}; x_{1,j}^{\mathbf{S}}, \dots, x_{n+1,j}^{\mathbf{S}})$ .
4. For  $l = 1, \dots, k$ , if  $|\mathcal{F}_l| + \mathcal{B}_l \geq t_b + 1$  then  $\mathbf{S}$  considers  $\mathcal{B}_l$  as an acceptable set, otherwise  $\mathbf{S}$  considers  $\mathcal{B}_l$  as unacceptable.

In [54], the authors claimed that there can be at most  $u$  acceptable sets, one corresponding to each wire in the bottom band. However, we now show that this is not true. In fact, we show that in the worst case there can be  $\mathcal{O}(3^{t_b})$  acceptable sets. Before doing so, we first present few concepts from graph theory.

**Definition 15.29 (Maximal Clique and Maximal Independent Set [1, 2])** *A maximal clique in a graph is a clique that cannot be extended by adding one more vertex to the clique. Complimentarily a maximal independent set is an independent set which cannot be extended by adding one more vertex to the independent set. If  $S$  is a maximal independent set in some graph, then it is a maximal clique in the complementary graph.*

**Definition 15.30 (Tuán Graph[3])** *The Tuán Graph  $T(n, r)$  is a graph formed by partitioning a set of  $n$  vertices into  $r$  subsets, with sizes as equal as possible, and connecting two vertices by an edge whenever they belong to different subsets. That is, it is a complete  $r$ -partite graph*

$$K_{\lceil n/r \rceil, \lceil n/r \rceil, \dots, \lfloor n/r \rfloor, \lfloor n/r \rfloor}$$

The following result from [47] states the upper bound on the number of maximal cliques that can be possible in any graph.

**Theorem 15.31 ([47, 1, 2])** *Any graph with  $n$  vertices can have at most  $3^{\frac{n}{3}}$  maximal cliques.*

The Tuán graph  $T(n, \lceil n/3 \rceil)$ , also called as **Moon-Moser** graph satisfies the bound given in the above theorem, as shown in the following example:

**Example 15.32 (Largest Number of Maximal Cliques Possible in a Graph)** *Let  $G$  be a graph with  $n$  vertices, which is a disjoint union of  $n/3$  triangle graphs. Any maximal independent set in this graph is formed by choosing one vertex from each triangle. It is easy to see that there will be  $3^{n/3}$  maximal independent sets in  $G$ . Moreover, these maximal independent sets will be maximal clique in the complementary graph  $\overline{G}$ . Thus  $\overline{G}$  will have exactly  $3^{n/3}$  maximal cliques. The graph  $\overline{G}$  is nothing, but the Tuán graph  $T(n, \lceil n/3 \rceil)$ , which is also called as **Moon-Moser** graph.*

Now we return back to the computation done by **S** at the end of **Phase II** in protocol  $\Pi_{modified}^{Existing}$ , as given in Fig. 15.6. We define the following graph:

**Definition 15.33 (Consistency Graph)** *Let  $G = (V, E)$  be an undirected graph where  $V = \{b_1, \dots, b_u\}$  and  $(b_i, b_j) \in E$  iff  $b_i, b_j$  are pairwise consistent and satisfies the following conditions:*

1.  $\beta_i^{\mathbf{S}} = \beta_j^{\mathbf{S}}$ ;
2.  $\alpha_{i,j}^{\mathbf{S}} = \text{hash}(\text{key}_{i,j}^{\mathbf{S}}; y_{1,i}^{\mathbf{S}}, \dots, y_{n+u,i}^{\mathbf{S}})$ ;
3.  $\alpha_{j,i}^{\mathbf{S}} = \text{hash}(\text{key}_{j,i}^{\mathbf{S}}; y_{1,j}^{\mathbf{S}}, \dots, y_{n+u,j}^{\mathbf{S}})$ .

*Then the graph  $G$  is called the consistency graph.*

We next claim that each  $\mathcal{B}_l$  computed by **S** in  $\Pi_{modified}^{Existing}$  is a maximal clique in the consistency graph  $G$ .

**Claim 15.34** *Each  $\mathcal{B}_l$  computed by **S** in Fig. 15.6 is a maximal clique in consistency graph  $G$ .*

PROOF: Follows from the definition of maximal clique, consistency graph and the steps executed to compute  $\mathcal{B}_l$ .  $\square$

**Claim 15.35** *Let  $b_i$  be a wire in the bottom band which is under the control of the adversary and let  $b_j$  be another wire in the bottom band (other than  $b_i$ ). Then irrespective of whether  $b_j$  is under the control of the adversary or not, the adversary can control the behavior of  $b_i$  during **Phase II** of  $\Pi_{modified}^{Existing}$  and decide whether  $b_i$  is consistent with  $b_j$  or not.*

PROOF: First of all, in order that  $b_i, b_j$  are pair-wise consistent, the following must hold:

1.  $\beta_i^{\mathbf{S}} = \beta_j^{\mathbf{S}}$ ;
2.  $\alpha_{i,j}^{\mathbf{S}} = \text{hash}(\text{key}_{i,j}^{\mathbf{S}}; y_{1,i}^{\mathbf{S}}, \dots, y_{n+u,i}^{\mathbf{S}})$ ;
3.  $\alpha_{j,i}^{\mathbf{S}} = \text{hash}(\text{key}_{j,i}^{\mathbf{S}}; y_{1,j}^{\mathbf{S}}, \dots, y_{n+u,j}^{\mathbf{S}})$ .

Since  $b_i$  is under the control of the adversary, the following information is completely under his control:

1.  $\beta_i^{\mathbf{S}}$ ;
2. The  $(n+u)$ -tuple  $(y_{1,i}^{\mathbf{S}}, \dots, y_{n+u,i}^{\mathbf{S}})$ ;
3. The pairs  $(\text{key}_{j,i}^{\mathbf{S}}; \alpha_{j,i}^{\mathbf{S}})$ , for all  $j = 1, \dots, u$ .

Now let  $b_j$  be a specific wire in the bottom band (other than  $b_i$ ). If  $b_j$  is also under the control of the adversary, then the adversary can always control  $b_i, b_j$  and decide whether  $b_i, b_j$  are consistent or inconsistent. On the other hand, even if  $b_j$  is honest, the adversary can control  $b_i$  and decide whether  $b_i$  and  $b_j$  are consistent or inconsistent. Specifically, if adversary wants that  $b_i$  and  $b_j$  are consistent, then the adversary does not modify the information passed over wire  $b_i$ . That is,  $\beta_i^{\mathbf{S}} = \beta_i^{\mathbf{R}}$ ,  $(y_{1,i}^{\mathbf{S}}, \dots, y_{n+u,i}^{\mathbf{S}}) = (y_{1,i}^{\mathbf{R}}, \dots, y_{n+u,i}^{\mathbf{R}})$  and  $(\text{key}_{j,i}^{\mathbf{S}}; \alpha_{j,i}^{\mathbf{S}}) = (\text{key}_{j,i}^{\mathbf{R}}; \alpha_{j,i}^{\mathbf{R}})$ . Since  $b_j$  is anyway honest, this implies that  $b_i, b_j$  will be pairwise consistent.

On the other hand, suppose adversary wants  $b_i, b_j$  to be inconsistent. The adversary can do so by arbitrarily changing  $(\text{key}_{j,i}^{\mathbf{S}}; \alpha_{j,i}^{\mathbf{S}})$ , such that  $(\text{key}_{j,i}^{\mathbf{S}}; \alpha_{j,i}^{\mathbf{S}}) \neq (\text{key}_{j,i}^{\mathbf{R}}; \alpha_{j,i}^{\mathbf{R}})$ . In this case,  $b_i, b_j$  will not be consistent.  $\square$

**Lemma 15.36** *The adversary can behave during Phase II of protocol  $\Pi_{\text{modified}}^{\text{Existing}}$  in such a way that it results in  $\mathcal{O}(3^{t_b})$  acceptable sets.*

PROOF: To prove the lemma, we consider the following specific network settings and adversarial behavior. However, the settings and the behavior can be easily generalized. Suppose  $n = t_b + 1$  and  $u = t_b$ . Moreover, without loss of generality, let  $f_1, \dots, f_{t_b}$  and  $b_{t_b}$  be the honest wires in the top band and bottom band respectively. Furthermore, let  $f_{t_b+1}$  and  $b_1, \dots, b_{t_b-1}$  be the wires under the control of the adversary in top band and bottom band respectively.

Now suppose that the adversary controls  $b_1, \dots, b_{t_b-1}$  in such a way that none of these wires are pairwise consistent with the honest wire  $b_{t_b}$ . That is, vertex  $b_{t_b}$  becomes an isolated vertex in the consistency graph. As explained in Claim 15.35, the adversary can always control  $b_1, \dots, b_{t_b-1}$  in such a way which causes this situation. Moreover, let the adversary controls  $b_1, \dots, b_{t_b-1}$  in such a way that the consistency graph induced by the vertex set  $\{b_1, \dots, b_{t_b-1}\}$  results in a Turán graph  $T(t_b - 1, \lceil (t_b - 1)/3 \rceil)$ . Again, since the wires  $b_1, \dots, b_{t_b-1}$  are under the control of the adversary, the adversary can control these wires so as to create the above situation. Now as stated in Theorem 15.31 and shown in Example 15.32, the graph  $T(t_b - 1, \lceil (t_b - 1)/3 \rceil)$  will have  $3^{t_b-1} = \mathcal{O}(3^{t_b})$  maximal cliques. Moreover, from Claim 15.34, each of these maximal cliques will be considered as a distinct  $\mathcal{B}_l$  by  $\mathbf{S}$ . Thus,  $\mathbf{S}$  will get  $\mathcal{O}(3^{t_b})$  distinct  $\mathcal{B}_l$ 's at the end of **Phase II**. Next we show that the adversary can control  $b_1, \dots, b_{t_b-1}$  in such a way that each of these  $\mathcal{O}(3^{t_b})$  distinct  $\mathcal{B}_l$ 's become valid acceptable sets.

Recall that during **Phase II**, **R** sends  $\beta^{\mathbf{R}}$  over the entire bottom band. Suppose that the adversary does not modify  $\beta^{\mathbf{R}}$  during its transmission over wires  $b_1, \dots, b_{t_b-1}$ . In this case, all the honest wires in the top band, namely  $f_1, \dots, f_{t_b}$  will be added in each  $\mathcal{F}_l$ . Since each  $\mathcal{B}_l$  contains at least one wire, it implies that  $|\mathcal{F}_l| + |\mathcal{B}_l| \geq t_b + 1$  will hold for all  $\mathcal{B}_l$ 's. Thus each  $\mathcal{B}_l$  will be considered as acceptable. Thus there will be  $\mathcal{O}(3^{t_b})$  acceptable sets.  $\square$

Since **Phase III** of protocol  $\Pi_{modified}^{Existing}$  is same as **Phase III** of protocol 3-SSMT-Static-Byzantine-Directed, where **S** tried to send  $m^{\mathbf{S}}$  using all possible acceptable sets, we have the following theorem:

**Theorem 15.37** *In protocol  $\Pi_{modified}^{Existing}$ , **S** and **R** may have to do exponential computation. Moreover, **S** may do exponential communication.*

PROOF: As explained in previous lemma, **S** and **R** may end up with exponential number of acceptable sets in protocol  $\Pi_{modified}^{Existing}$ . Thus they have to do exponential computation. It is easy to see that in this case, **S** has to do exponential communication during **Phase III**, as **S** has to send  $m^{\mathbf{S}}$ , corresponding to each acceptable set by using the keys computed from it.  $\square$

Now as in protocol  $\Pi_{modified}^{Existing}$ , the adversary may control the bottom band in such a way that **S** may end up with exponential number of acceptable sets at the end of **Phase II** of the three phase efficient SSMT protocol  $\Pi^{Existing}$  of [24, 87]. We capture this by the following theorem statement.

**Theorem 15.38** *In protocol  $\Pi^{Existing}$ , **S** and **R** may have to do exponential computation. Moreover, **S** may do exponential communication.*

Since protocol  $\Pi_{modified}^{Existing}$  is used as a black-box in other SRMT and SSMT protocols of [54], it will make the computational and communication complexity of all SRMT and SSMT protocols of [54] exponential.

**Remark 15.39 (Difference Between 3-SSMT-Static-Byzantine-Directed and  $\Pi_{modified}^{Existing}$ )**  
*The main difference between **Phase II** of protocol 3-SSMT-Static-Byzantine-Directed and  $\Pi_{modified}^{Existing}$  is the way **S** divides the bottom band. In  $\Pi_{modified}^{Existing}$ , it is required that all the wires in a  $\mathcal{B}_l$  should be pairwise consistent, thus making  $\mathcal{B}_l$  a maximal clique. On the other hand, in 3-SSMT-Static-Byzantine-Directed, it is required that all the wires in a  $\mathcal{B}_l$  should be pairwise consistent only with wire  $b_l$ . It is this subtle difference which results in at most  $u$  acceptable sets (one corresponding to each wire in the bottom band) in protocol 3-SSMT-Static-Byzantine-Directed.*

Though protocol 3-SSMT-Static-Byzantine-Directed is an efficient SSMT protocol, it is not a communication optimal protocol. We can further reduce the communication complexity of SSMT protocols by increasing the number of phases in the protocol, which will further lead us to the design of a communication optimal SSMT protocol. In order to design our communication optimal protocol, we require few more black box, which we describe in the subsequent sections.

## 15.6 Six Phase Statistically Secure Pad Establishment Protocol

We now propose a six phase protocol called 6-Pad, which correctly establishes a random non-zero one time pad between **S** and **R** with very high probability by communicating

$\mathcal{O}(n^3)$  field elements. Moreover, the pad will be information theoretically secure. If the entire bottom band is corrupted, then the size of the pad is  $\Theta(n^2u)$  field elements. Otherwise the size of the pad is  $\Theta(n^2)$  field elements. Before presenting protocol 6-Pad, we present another protocol called 1-Pad, which will be used as a black-box in protocol 6-Pad.

### 15.6.1 Single Phase Conditional Pad Establishment Protocol

Suppose **S** and **R** somehow in advance knows that full bottom band is corrupted. This implies that at most  $t_b - u$  wires in the top band are corrupted. This further implies that there exists at least  $t_b + 1$  honest wires in the top band. Under this assumption, we design a single phase protocol called 1-Pad which allows **S** and **R** to correctly establish a non-zero random one time pad of size  $\Theta(n^2u)$  with very high probability by communicating  $\mathcal{O}(n^3)$  field elements. Moreover, the pad will be information theoretically secure.

The idea of the protocol is similar to the one used in **Phase I** of protocol 3-SSMT-Static-Byzantine-Directed. Recall that **Phase I** of protocol 3-SSMT-Static-Byzantine-Directed would be successful if there exists at least  $t_b + 1$  honest wires in the top band (see Lemma 15.10). Now in 1-Pad we have  $t_b + 1$  honest wires in the top band. So if we execute **Phase I** of 3-SSMT-Static-Byzantine-Directed then it would be successful. Protocol 1-Pad is based on this principle. The protocol uses Extrapolation Technique (see Fig. 8.1). Protocol 1-Pad is formally given in Fig. 15.7.

We now prove the properties of protocol 1-Pad.

**Claim 15.40** *In protocol 1-Pad if **R** concludes that  $F_i^{\mathbf{R}}$  is a valid row of  $B^{\mathbf{S}}$  then except with probability  $2^{-\Omega(\kappa)}$ ,  $F_i^{\mathbf{R}} = F_i^{\mathbf{S}}$ .*

PROOF: The proof follows using similar argument as in Claim 15.6.  $\square$

**Claim 15.41** *In protocol 1-Pad,  $\mathcal{P}^{\mathbf{S}}$  will be information theoretically secure.*

PROOF: Recall that in protocol 1-Pad, there are at most  $t_b - u$  wires in the top band which can be under the control of the adversary. Now using similar argument as in Claim 15.7, it follows that  $[(t_b + 1) - (t_b - u)]n^2 = (u + 1)n^2$  elements of  $\mathcal{V}^{\mathbf{S}}$  will be information theoretically secure. The rest now follows from the properties of algorithm EXTRAND.  $\square$

**Claim 15.42** *If **R** gets  $t_b + 1$  valid rows of  $B^{\mathbf{S}}$  then **R** can recover  $\mathcal{P}^{\mathbf{S}}$ .*

PROOF: Follows using similar argument as in Claim 15.8.  $\square$

**Claim 15.43** *If **R** outputs  $\mathcal{P}^{\mathbf{R}}$  then except with error probability  $2^{-\Omega(\kappa)}$ ,  $\mathcal{P}^{\mathbf{R}} = \mathcal{P}^{\mathbf{S}}$ .*

PROOF: If **R** outputs  $\mathcal{P}^{\mathbf{R}}$ , then it implies that **R** has received  $t_b + 1$  valid rows. From Claim 15.40, all these rows are indeed the rows of  $B^{\mathbf{S}}$  sent by **S** with very high probability. So from Claim 15.42, with very high probability,  $\mathcal{P}^{\mathbf{R}} = \mathcal{P}^{\mathbf{S}}$ .  $\square$

**Theorem 15.44** *If the entire bottom band is corrupted, then protocol 1-Pad securely establishes a random non-zero pad of size  $\Theta(n^2u\kappa)$  bits by communicating  $\mathcal{O}(n^3\kappa)$  bits.*

PROOF: Over each wire, **S** sends  $\mathcal{O}(n^2)$  field elements. This incurs a communication complexity of  $\mathcal{O}(n^3)$  field elements and hence  $\mathcal{O}(n^3\kappa)$  bits. The rest of the properties of protocol 1-Pad follows from Claim 15.40, Claim 15.41, Claim 15.42 and Claim 15.43.  $\square$

Figure 15.7: Single Phase Protocol 1-Pad

**Computation and Communication by S:**

1. **S** sets  $ROW = t_b + 1, COL = c = n^2 + t_b - u$  and  $N = n$ . **S** then forms a matrix  $B_{init}$ , consisting of  $ROW \times COL$  random, non-zero elements from  $\mathbb{F}$ . **S** then executes the steps in Extrapolation Technique to generate an  $n \times c$  matrix  $B_{ext}$  from  $B_{init}$ . Let  $B^{\mathbf{S}} = B_{ext}$ .
2. For  $i = 1, \dots, n$ , **S** selects a random non-zero hash key  $\alpha_i^{\mathbf{S}}$ , corresponding to wire  $f_i$ .
3. For  $i = 1, \dots, n$ , **S** sends the following to **R** over wire  $f_i$ :
  - (a) The  $i^{th}$  row of  $B^{\mathbf{S}}$ , denoted by  $F_i^{\mathbf{S}}$ ;
  - (b) The hash key  $\alpha_i^{\mathbf{S}}$  and
  - (c) The hash values  $v_{ji}^{\mathbf{S}}$ , where  $v_{ji}^{\mathbf{S}} = hash(\alpha_i^{\mathbf{S}}; F_j^{\mathbf{S}})$ , for  $j = 1, \dots, n$ .
4. Let  $\mathcal{V}^{\mathbf{S}}$  denote the concatenation of the elements of the first  $t_b + 1$  rows of  $B^{\mathbf{S}}$ . **S** computes
$$\mathcal{P}^{\mathbf{S}} = \text{EXTRAND}_{|\mathcal{V}^{\mathbf{S}}|, (u+1)n^2}(\mathcal{V}^{\mathbf{S}}).$$
5. The vector  $\mathcal{P}^{\mathbf{S}}$  denotes the information theoretically secure random pad of size  $\Theta(n^2u)$  which will be correctly established with **R** with very high probability.

**Computation by R:**

1. For  $i = 1, \dots, n$ , let **R** receive the following over wire  $f_i$ :
  - (a) The  $c$ -tuple, denoted by  $F_i^{\mathbf{R}}$ ;
  - (b) The hash key  $\alpha_i^{\mathbf{R}}$  and
  - (c) The hash values  $v_{ji}^{\mathbf{R}}$ , for  $j = 1, \dots, n$ .
2. For  $i = 1, \dots, n$ , **R** computes
$$\text{Support}_i = |\{j : hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{R}}) = v_{ij}^{\mathbf{R}}\}|.$$
3. If  $\text{Support}_i \geq t_b + 1$ , then **R** concludes that  $F_i^{\mathbf{R}}$  is a valid row of  $B^{\mathbf{S}}$ . Otherwise, **R** concludes that  $F_i^{\mathbf{R}}$  is an invalid row.
4. Using  $t_b + 1$  valid rows, **R** constructs the  $n \times c$  array  $B^{\mathbf{R}}$ . From  $B^{\mathbf{R}}$ , **R** computes  $\mathcal{V}^{\mathbf{R}}$ , from which it finally computes  $\mathcal{P}^{\mathbf{R}}$  and terminates.

### 15.6.2 A Six Phase Pad Establishment Protocol

We now present our six phase pad establishment protocol 6-Pad. The main idea of the protocol is as follows: **S** and **R** interacts to find whether there the entire bottom band is corrupted or not. If they find that the entire bottom band is corrupted then **S** and **R** executes the single phase protocol 1-Pad to establish a pad of size  $\Theta(n^2u)$

field elements. On the other hand if **S** and **R** finds that the entire bottom band is not corrupted then they apply EXTRAND on the information exchanged between them to establish a pad of size  $\Theta(n^2)$  field elements.

Now as in the case of protocol 3-SSMT-Static-Byzantine-Directed, we present protocol 6-Pad phase by phase. This will help the reader to understand the protocol conceptually. We begin with the description of the first two phases.

### 15.6.2.1 First Two Phases of Protocol 6-Pad

The first two phases of protocol 6-Pad are similar as in protocol 3-SSMT-Static-Byzantine-Directed, except that now the first phase is initiated by **R**. During first phase, **R** sends a random  $(n^2 + 1)$ -tuple to **S** over each wire. During second phase, **S** sends a random  $(n^2 + t_b)$ -tuple to **R** over each wire. In addition to this, **S** and **R** also exchanges the hash values of the exchanged tuples, as in protocol 3-SSMT-Static-Byzantine-Directed. The formal details of first two phases of 6-Pad are given in Fig. 15.8.

At the end of **Phase II** of protocol 6-Pad, **R** divides the top band using similar principle as used by **S** to divide the bottom band during protocol 3-SSMT-Static-Byzantine-Directed. We now state the following lemmas, whose proofs are similar to the one given for protocol 3-SSMT-Static-Byzantine-Directed.

**Lemma 15.45** *Let  $f_i$  be an honest wire in the top band. Then  $\mathcal{F}_i$  and corresponding  $\mathcal{B}_i$  will have the following properties:*

1.  $\mathcal{F}_i$  will be an acceptable set.
2. All honest wires in the bottom band will be present in  $\mathcal{B}_i$ , while all the honest wires in the top band will be present in  $\mathcal{F}_i$ .
3. With very high probability, the  $(n^2 + t_b)$ -tuple received by **R** at the end of **Phase II** over the wires in  $\mathcal{F}_i$  are not modified.
4. With very high probability, the  $(n^2 + 1)$ -tuple received by **S** at the end of **Phase I** over the wires in  $\mathcal{B}_i$  are not modified.
5. The adversary will have no information about the first  $n^2$  values of the  $(n^2 + t_b)$ -tuples which are exchanged over the honest wire(s) in  $\mathcal{F}_i$ . The adversary will also have no information about the first  $n^2$  values of the  $(n^2 + 1)$ -tuples which are exchanged over the honest wires in  $\mathcal{B}_i$ .

**Lemma 15.46** *Let  $f_i$  be a wire in the top band such that  $f_i$  is under the control of the adversary. Moreover, let  $\mathcal{F}_i$  be an acceptable set. Then  $\mathcal{F}_i$  and corresponding  $\mathcal{B}_i$  will have the following properties:*

1. There will exist at least one honest wire, either from the top band or bottom band, which will be present in  $\mathcal{F}_i$  or  $\mathcal{B}_i$  respectively.
2. If there exists an honest wire in  $\mathcal{F}_i$ , then the  $(n^2 + t_b)$ -tuple is correctly exchanged between **S** and **R** over that wire. Moreover, adversary will have no information about the first  $n^2$  values of the  $(n^2 + t_b)$ -tuple.
3. If there exists an honest wire in  $\mathcal{B}_i$ , then the  $(n^2 + 1)$ -tuple is correctly exchanged between **S** and **R** over that wire. Moreover, adversary will have no information about the first  $n^2$  values of the  $(n^2 + 1)$ -tuple.

Figure 15.8: First Two Phases of Protocol 6-Pad

**Phase I: R to S:**

1. For  $i = 1, \dots, u$ , **R** selects a random non-zero  $(n^2 + 1)$ -tuple  $(y_{1,i}^{\mathbf{R}}, \dots, y_{n^2+1,i}^{\mathbf{R}})$ , corresponding to wire  $b_i$  and sends  $(y_{1,i}^{\mathbf{R}}, \dots, y_{n^2+1,i}^{\mathbf{R}})$  to **S** over wire  $b_i$ .

**Phase II: S to R:**

1. Let **S** receive  $(y_{1,i}^{\mathbf{S}}, \dots, y_{n^2+1,i}^{\mathbf{S}})$  along wire  $b_i$ , for  $i = 1, \dots, u$ .
2. For  $i = 1, \dots, u$ , **S** selects a random non-zero hash key  $r_i^{\mathbf{S}}$ , corresponding to wire  $b_i$ .
3. **S** computes the set  $\beta^{\mathbf{S}} = \{(r_i^{\mathbf{S}}, \gamma_i^{\mathbf{S}}) : i = 1, \dots, u\}$ , where  $\gamma_i^{\mathbf{S}} = \text{hash}(r_i^{\mathbf{S}}; y_{1,i}^{\mathbf{S}}, \dots, y_{n^2+1,i}^{\mathbf{S}})$ .
4. **S** associates a random non-zero  $(n^2 + t_b)$ -tuple  $(x_{1,i}^{\mathbf{S}}, \dots, x_{n^2+t_b,i}^{\mathbf{S}})$  with wire  $f_i$ , for  $i = 1, \dots, n$ .
5. For  $i = 1, \dots, n$ , corresponding to wire  $f_i$ , **S** chooses  $n$  random, non-zero hash key  $key_{i,j}^{\mathbf{S}}$ , for  $j = 1, \dots, n$ .
6. For  $i = 1, \dots, n$ , **S** sends the following to **R** over wire  $f_i$ :
  - (a)  $\beta^{\mathbf{S}}$ ;
  - (b) The  $(n^2 + t_b)$ -tuple  $(x_{1,i}^{\mathbf{S}}, \dots, x_{n^2+t_b,i}^{\mathbf{S}})$ ;
  - (c) The pairs  $\{(key_{j,i}^{\mathbf{S}}, \alpha_{j,i}^{\mathbf{S}}) : j = 1, \dots, n\}$ , where  $\alpha_{j,i}^{\mathbf{S}} = \text{hash}(key_{j,i}^{\mathbf{S}}; x_{1,j}^{\mathbf{S}}, \dots, x_{n^2+t_b,j}^{\mathbf{S}})$ .

**Computation by R at the end of Phase II:**

1. Let **R** receive the following over wire  $f_i$ , for  $i = 1, \dots, n$ :
  - (a)  $\beta_i^{\mathbf{S}} = \{(r_{i,j}^{\mathbf{R}}, \gamma_{i,j}^{\mathbf{R}}) : j = 1, \dots, u\}$ ;
  - (b) The  $(n^2 + t_b)$ -tuple  $(x_{1,i}^{\mathbf{R}}, \dots, x_{n^2+t_b,i}^{\mathbf{R}})$ ;
  - (c) The pairs  $\{(key_{j,i}^{\mathbf{R}}, \alpha_{j,i}^{\mathbf{R}}) : j = 1, \dots, n\}$
2. For  $i = 1, \dots, n$ , corresponding to wire  $f_i$ , **S** computes the set  $\mathcal{F}_i$  and  $\mathcal{B}_i$  as follows:
  - (a) **R** adds wire  $f_j \in \{f_1, \dots, f_n\}$  to  $\mathcal{F}_i$  (which is initially  $\emptyset$ ) if  $f_i, f_j$  are found to be pairwise consistent by satisfying the following conditions:
    - i.  $\alpha_{i,j}^{\mathbf{R}} = \text{hash}(key_{i,j}^{\mathbf{R}}; x_{1,i}^{\mathbf{R}}, \dots, x_{n^2+t_b,i}^{\mathbf{R}})$  and
    - ii.  $\alpha_{j,i}^{\mathbf{R}} = \text{hash}(key_{j,i}^{\mathbf{R}}; x_{1,j}^{\mathbf{R}}, \dots, x_{n^2+t_b,j}^{\mathbf{R}})$ .
  - (b) **R** adds wire  $b_j$  to  $\mathcal{B}_i$  (which is initially  $\emptyset$ ) if  $\gamma_{i,j}^{\mathbf{R}} = \text{hash}(r_{i,j}^{\mathbf{R}}; y_{1,j}^{\mathbf{R}}, \dots, y_{n^2+1,j}^{\mathbf{R}})$ .
  - (c) If  $|\mathcal{F}_i| + |\mathcal{B}_i| \geq t_b + 1$  then **R** considers  $\mathcal{F}_i$  as an acceptable set. Otherwise **R** considers  $\mathcal{F}_i$  as unacceptable.

4. If there exists an honest wire in  $\mathcal{F}_i$ , then with very high probability the  $(n^2 + t_b)$ -tuple is correctly exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  over wire  $f_i$ . However, the adversary can modify the  $(n^2 + t_b)$ -tuples which are exchanged over the corrupted wires in  $\mathcal{F}_i$ , other than  $f_i$ , without letting  $\mathbf{S}$  and  $\mathbf{R}$  know about it.
5. If there is no honest wire in  $\mathcal{F}_i$ , then the adversary can always modify the  $(n^2 + t_b)$ -tuples which are exchanged over the corrupted wires in  $\mathcal{F}_i$ , without letting  $\mathbf{S}$  and  $\mathbf{R}$  know about it.
6. Irrespective of the number of honest wires in the bottom band, the adversary can always modify the  $(n^2 + 1)$ -tuple which is exchanged over a corrupted wire (if any) in  $\mathcal{B}_i$ , without letting  $\mathbf{S}$  and  $\mathbf{R}$  know about it.

**Lemma 15.47** *In protocol 6-Pad, there can be at most  $n$  acceptable sets.*

**Lemma 15.48** *In protocol 6-Pad,  $\mathbf{R}$  communicates  $\mathcal{O}(n^2u)$  field elements during Phase I and  $\mathbf{S}$  communicates  $\mathcal{O}(n^3)$  field elements during Phase II.*

We now proceed to the description of the remaining phases of the protocol 6-Pad which are given in the next subsection.

### 15.6.2.2 Remaining Phases of Protocol 6-Pad

Corresponding to each acceptable set  $\mathcal{F}_l$ ,  $\mathbf{R}$  concatenates the tuples which are exchanged over the wires in  $\mathcal{F}_l$  and  $\mathcal{B}_l$ .  $\mathbf{R}$  then hashes the resultant tuple and sends the hash value, along with the identity of  $\mathcal{F}_l$  and  $\mathcal{B}_l$  to  $\mathbf{S}$  through all the wires in  $\mathcal{B}_l$ . If the entire bottom band is corrupted, then  $\mathbf{S}$  will not receive the correct hash value and with very high probability,  $\mathbf{S}$  will come to know this. This is because there will exist at least one honest wire, either in  $\mathcal{F}_l$  and  $\mathcal{B}_l$ , such that adversary will have no information about the first  $n^2$  element of the tuple exchanged over the honest wire. In this case,  $\mathbf{S}$  notifies about his finding to  $\mathbf{R}$  and then executes protocol 1-Pad to establish a pad of size  $\Theta(n^2u)$ .

On the other hand, if there exists at least one honest wire in the bottom band, then  $\mathbf{S}$  will correctly receive a valid hash value, along with the identity of corresponding  $\mathcal{F}_l$  and  $\mathcal{B}_l$ .  $\mathbf{S}$  then notifies  $\mathbf{R}$  about the identity of  $\mathcal{F}_l$  and  $\mathcal{B}_l$ .  $\mathbf{S}$  also applies EXTRAND on the tuples which are exchanged along the wires in  $\mathcal{F}_l$  and  $\mathcal{B}_l$  to extract an information theoretically secure pad of size  $n^2$ . Application of EXTRAND is required because it may happen that  $\mathcal{F}_l$  and  $\mathcal{B}_l$  contains  $t_b + 1$  wires in total, out of which  $t_b$  are under the control of the adversary. So there will be only one honest wire and the adversary will not know the first  $n^2$  elements of the tuple which is exchanged over that honest wire. Since  $\mathbf{S}$  and  $\mathbf{R}$  will not know the exact identity of the honest wire, they apply EXTRAND.

Notice that it is not easy for  $\mathbf{S}$  to notify  $\mathbf{R}$  about its finding. This is because there can be only  $t_b + 1$  wires in the top band and in the worst case,  $t_b$  of them can be corrupted. To deal with this problem,  $\mathbf{S}$  notifies  $\mathbf{R}$  about its finding using protocol 3-SSMT-Static-Byzantine-Directed. Since 3-SSMT-Static-Byzantine-Directed is an SSMT protocol, it will securely and hence correctly deliver the notification to  $\mathbf{R}$ . The formal details of the remaining phases of protocol 6-Pad are given in Fig. 15.9.

We now prove the properties of remaining phases of protocol 6-Pad, as given in Fig. 15.9.

**Claim 15.49** *Let  $\mathcal{F}_l$  be an acceptable set. Then adversary will have no information about  $n^2 - 1$  elements of  $\mathcal{V}_l^{\mathbf{R}}$ .*

Figure 15.9: Remaining Phases of Protocol 6-Pad

**Phase III: R to S:** For each acceptable set  $\mathcal{F}_l$  and the corresponding set  $\mathcal{B}_l$ , **R** does the following:

1. Let  $\mathcal{V}_l^{\mathbf{R}}$  denote the concatenation of the first  $n^2$  elements from  $(n^2 + 1)$ -tuples and  $(n^2 + t_b)$ -tuples, which **R** has sent and received over the wires in  $\mathcal{B}_l$  and  $\mathcal{F}_l$  respectively.
2. Corresponding to vector  $\mathcal{V}_l^{\mathbf{R}}$ , **R** selects a random non-zero hash key  $\mathcal{K}_l^{\mathbf{R}}$ .
3. **R** computes  $\delta_l^{\mathbf{R}} = \text{hash}(\mathcal{K}_l^{\mathbf{R}}; \mathcal{V}_l^{\mathbf{R}})$  and sends  $\mathcal{B}_l, \mathcal{F}_l$  and the 2-tuple  $(\mathcal{K}_l^{\mathbf{R}}, \delta_l^{\mathbf{R}})$  to **S** through all the wires in  $\mathcal{B}_l$ .

**Computation by S at the end of Phase III:** Now using the hash value(s) received from **R**, the sender **S** tries to find whether there exists at least one uncorrupted wire in the bottom band. For this, **S** does the following:

1. Let **S** receive the index set  $\mathcal{F}_{i,l}^{\mathbf{S}}$  and  $\mathcal{B}_{i,l}^{\mathbf{S}}$  and the 2-tuple  $(\mathcal{K}_{i,l}^{\mathbf{S}}, \delta_{i,l}^{\mathbf{S}})$  along wire  $b_i$ , for  $i = 1, \dots, u$ . Here  $l \leq n$ .
2. If for some  $i \in \{1, \dots, u\}$  and some  $l \leq n$ ,  $|\mathcal{F}_{i,l}^{\mathbf{S}}| + |\mathcal{B}_{i,l}^{\mathbf{S}}| \leq t_b$ , then **S** concludes that wire  $b_i$  is corrupted and neglects it.
3. If  $b_i$  is not neglected during previous step then for each  $\mathcal{F}_{i,l}^{\mathbf{S}}, \mathcal{B}_{i,l}^{\mathbf{S}}$  and the tuple  $(\mathcal{K}_{i,l}^{\mathbf{S}}, \delta_{i,l}^{\mathbf{S}})$  received along wire  $b_i$ , **S** does the following:
  - (a) Let  $\mathcal{V}_{i,l}^{\mathbf{S}}$  denote the concatenation of first  $n^2$  values of the  $(n^2 + 1)$ -tuples and  $(n^2 + t_b)$ -tuples, which **S** has received and sent over the wires in  $\mathcal{B}_{i,l}^{\mathbf{S}}$  and  $\mathcal{F}_{i,l}^{\mathbf{S}}$  respectively.
  - (b) **S** now checks  $\delta_{i,l}^{\mathbf{S}} \stackrel{?}{=} \text{hash}(\mathcal{K}_{i,l}^{\mathbf{S}}; \mathcal{V}_{i,l}^{\mathbf{S}})$ .
  - (c) If the test fails for all received  $\mathcal{F}_{i,l}^{\mathbf{S}}, \mathcal{B}_{i,l}^{\mathbf{S}}$  and the tuple  $(\mathcal{K}_{i,l}^{\mathbf{S}}, \delta_{i,l}^{\mathbf{S}})$  then **S** concludes that wire  $b_i$  is corrupted and neglects all the values received along  $b_i$ .
  - (d) If the test succeeds for some  $\mathcal{F}_{i,l}^{\mathbf{S}}, \mathcal{B}_{i,l}^{\mathbf{S}}$  and the tuple  $(\mathcal{K}_{i,l}^{\mathbf{S}}, \delta_{i,l}^{\mathbf{S}})$  then then **S** does the following:
    - i. **S** concludes that the tuples are correctly exchanged between **S** and **R** along the wires in  $\mathcal{B}_{i,l}^{\mathbf{S}}$  and  $\mathcal{F}_{i,l}^{\mathbf{S}}$ .
    - ii. **S** applies EXTRAND on  $\mathcal{V}_{i,l}^{\mathbf{S}}$  to generate a vector  $\mathcal{P}_1^{\mathbf{S}}$  of size  $n^2 - 1$ .
    - iii. Finally **S** terminates 6-Pad by sending a special predefined "success" signal, along with the index of the wires in the set  $\mathcal{B}_{i,l}^{\mathbf{S}}$  and  $\mathcal{F}_{i,l}^{\mathbf{S}}$  to **R** by executing the protocol 3-SSMT-Static-Byzantine-Directed.
    - iv. At the end of 3-SSMT-Static-Byzantine-Directed, **R** securely (and hence correctly) receives the set  $\mathcal{B}_{i,l}^{\mathbf{S}}$  and  $\mathcal{F}_{i,l}^{\mathbf{S}}$ , computes  $\mathcal{P}_1^{\mathbf{R}}$  and terminates 6-Pad. Since 3-SSMT-Static-Byzantine-Directed takes three phases, **R** will terminate 6-Pad at the end of **Phase VI**.
4. If all the wires in the bottom band get discarded, then **S** concludes that entire bottom band is corrupted. In this case, **S** does the following:
  - (a) **S** sends a special predefined "failure" signal to **R** by executing the three phase protocol 3-SSMT-Static-Byzantine-Directed.
  - (b) Parallely, **S** establishes a secure pad  $\mathcal{P}_2^{\mathbf{S}}$  of size  $\Theta(n^2 u)$  field elements with **R** by executing single phase Protocol 1-Pad.
  - (c) At the end of 3-SSMT-Static-Byzantine-Directed, **R** will know that the entire bottom band is corrupted.
  - (d) Parallely at the end of 1-Pad, **R** will output the pad  $\mathcal{P}_2^{\mathbf{R}}$  of size  $\Theta(n^2 u)$  field elements. Since 3-SSMT-Static-Byzantine-Directed takes three phases, **R** will terminate 6-Pad at the end of **Phase VI**.

PROOF: Since  $\mathcal{F}_l$  is an acceptable set, it implies that there exists at least one honest wire, either in  $\mathcal{F}_l$  or corresponding set  $\mathcal{B}_l$ . Moreover, from Lemma 15.45 and Lemma 15.46, adversary will have no information about the first  $n^2$  elements of the tuple which is exchanged over that honest wire. So  $\mathcal{V}_l^{\mathbf{R}}$ , which is the concatenation of the first  $n^2$  elements from the tuples which are exchanged between  $\mathbf{S}$  and  $\mathbf{R}$  along the wires in  $\mathcal{F}_l$  and  $\mathcal{B}_l$  will have  $n^2$  elements which will be unknown to the adversary. Now during **Phase III**,  $\mathbf{R}$  sends one hash value of  $\mathcal{V}_l^{\mathbf{R}}$  along the wires in  $\mathcal{B}_l$ . So from the properties of hashing, it still holds that  $n^2 - 1$  elements of  $\mathcal{V}_l^{\mathbf{R}}$  are information theoretically secure.  $\square$

**Claim 15.50** *If at all  $\mathbf{S}$  computes a pad  $\mathcal{P}_1^{\mathbf{S}}$  of size  $n^2 - 1$  field elements, then  $\mathbf{R}$  will correctly output  $\mathcal{P}_1^{\mathbf{R}} = \mathcal{P}_1^{\mathbf{S}}$ , except with probability  $2^{-\Omega(\kappa)}$ .*

PROOF: Suppose  $\mathbf{S}$  computes  $\mathcal{P}_1^{\mathbf{S}}$  from  $\mathcal{F}_{i,l}^{\mathbf{S}}$ ,  $\mathcal{B}_{i,l}^{\mathbf{S}}$  and the tuple  $(\mathcal{K}_{i,l}^{\mathbf{S}}, \delta_{i,l}^{\mathbf{S}})$ . This implies that  $\mathcal{V}_{i,l}^{\mathbf{S}}$  computed from the tuples exchanged over the wires in  $\mathcal{F}_{i,l}^{\mathbf{S}}$  and  $\mathcal{B}_{i,l}^{\mathbf{S}}$  satisfies  $\delta_{i,l}^{\mathbf{S}} = \text{hash}(\mathcal{K}_{i,l}^{\mathbf{S}}; \mathcal{V}_{i,l}^{\mathbf{S}})$ . Now from the proof of the previous claim,  $n^2 - 1$  elements of  $\mathcal{V}_{i,l}^{\mathbf{S}}$  are information theoretically secure. So from the properties of hashing, it implies that the tuples are exchanged correctly between  $\mathbf{S}$  and  $\mathbf{R}$  along the wires in  $\mathcal{F}_{i,l}^{\mathbf{S}}$  and  $\mathcal{B}_{i,l}^{\mathbf{S}}$ , except with probability  $2^{-\Omega(\kappa)}$ . Now  $\mathbf{S}$  communicates the identity of the wires in  $\mathcal{F}_{i,l}^{\mathbf{S}}$  and  $\mathcal{B}_{i,l}^{\mathbf{S}}$  to  $\mathbf{R}$  by executing protocol 3-SSMT-Static-Byzantine-Directed. From the properties of 3-SSMT-Static-Byzantine-Directed,  $\mathbf{R}$  will correctly receive the identity of the wires in  $\mathcal{F}_{i,l}^{\mathbf{S}}$  and  $\mathcal{B}_{i,l}^{\mathbf{S}}$  except with probability  $2^{-\Omega(\kappa)}$ . So  $\mathcal{P}_1^{\mathbf{R}} = \mathcal{P}_1^{\mathbf{S}}$ , except with probability  $2^{-\Omega(\kappa)}$ .  $\square$

**Claim 15.51** *If at all  $\mathbf{S}$  computes a pad  $\mathcal{P}_2^{\mathbf{S}}$  of size  $\Theta(n^2u)$  field elements, then  $\mathbf{R}$  will correctly output  $\mathcal{P}_2^{\mathbf{R}} = \mathcal{P}_2^{\mathbf{S}}$ , except with probability  $2^{-\Omega(\kappa)}$ .*

PROOF: The reason that  $\mathbf{S}$  computes a pad  $\mathcal{P}_2^{\mathbf{S}}$  of size  $\Theta(n^2u)$  field elements is that all the wires in bottom band get rejected by  $\mathbf{S}$  at the end of **Phase III**. This implies that  $\mathbf{S}$  finds the entire bottom band to be corrupted, which  $\mathbf{S}$  notifies to  $\mathbf{R}$  by sending "failure" signal to  $\mathbf{R}$  by executing protocol 3-SSMT-Static-Byzantine-Directed. From the properties of 3-SSMT-Static-Byzantine-Directed,  $\mathbf{R}$  will correctly receive the "failure" signal and concludes that the entire bottom band is corrupted, except with error probability  $2^{-\Omega(\kappa)}$ . Now to establish the pad  $\mathcal{P}_2^{\mathbf{S}}$ ,  $\mathbf{S}$  executes the single phase protocol 1-Pad. So from the properties of 1-Pad,  $\mathbf{R}$  will correctly output  $\mathcal{P}_2^{\mathbf{R}} = \mathcal{P}_2^{\mathbf{S}}$  at the end of 1-Pad, except with probability  $2^{-\Omega(\kappa)}$ .  $\square$

**Claim 15.52** *Irrespective of whether  $\mathbf{S}$  and  $\mathbf{R}$  agrees on a pad of size  $n^2 - 1$  or  $\Theta(n^2u)$ , the pad will be information theoretically secure.*

PROOF: If  $\mathbf{S}$  and  $\mathbf{R}$  agrees on a pad of size  $n^2 - 1$ , it implies that the pad is computed by applying EXTRAND on some  $\mathcal{V}_{i,l}^{\mathbf{S}}$ . From the proof of Claim 15.50 and Claim 15.49, at least  $n^2 - 1$  elements of  $\mathcal{V}_{i,l}^{\mathbf{S}}$  are information theoretically secure. So from the properties of EXTRAND, the computed pad of size  $n^2 - 1$  will be information theoretically secure.

On the other hand if the agreed pad is of size  $\Theta(n^2u)$  then it implies that the pad is established by executing the protocol 1-Pad. In this case, security of the pad follows from the security of protocol 1-Pad.  $\square$

**Claim 15.53** *In the steps given in Fig. 15.9,  $\mathbf{S}$  and  $\mathbf{R}$  has to communicate  $\mathcal{O}(n^3)$  field elements.*

PROOF: During **Phase III**, corresponding to an acceptable set  $\mathcal{F}_l$ ,  $\mathbf{R}$  has to send the index of the wires in  $\mathcal{F}_l, \mathcal{B}_l$  and the tuple  $(\mathcal{K}_l^{\mathbf{R}}, \delta_l^{\mathbf{R}})$  through all the wires in  $\mathcal{B}_l$ . This requires a communication complexity of  $\mathcal{O}(nu)$  field elements. Since there can be  $n$  acceptable sets, it implies that  $\mathbf{R}$  has to communicate  $\mathcal{O}(n^2u) = \mathcal{O}(n^3)$  field elements during **Phase III**. During **Phase IV**,  $\mathbf{S}$  will execute protocol 3-SSMT-Static-Byzantine-Directed to notify either "success" or "failure" signal to  $\mathbf{R}$ . From Theorem 15.28, this requires a communication complexity of  $\mathcal{O}(n^3)$  field elements. Thus  $\mathbf{S}$  and  $\mathbf{R}$  has to communicate  $\mathcal{O}(n^3)$  field elements in the steps given in Fig. 15.9.  $\square$

**Claim 15.54** *Protocol 6-Pad terminates in six phases.*

PROOF: Follows from the steps given in Fig. 15.8 and Fig. 15.9.  $\square$

**Theorem 15.55 (Properties of Protocol 6-Pad)** *Protocol 6-Pad has the following properties:*

1. *If the entire bottom band is corrupted, then  $\mathbf{S}$  and  $\mathbf{R}$  correctly establishes a pad of size  $\Theta(n^2u\kappa)$  bits in six phases by communicating  $\mathcal{O}(n^3\kappa)$  bits, except with error probability  $2^{-\Omega(\kappa)}$ . Moreover, the pad will be information theoretically secure.*
2. *If there exists at least one honest wire in the bottom band, then  $\mathbf{S}$  and  $\mathbf{R}$  correctly establishes a pad of size  $\Theta(n^2\kappa)$  bits in six phases by communicating  $\mathcal{O}(n^3\kappa)$  bits, except with error probability  $2^{-\Omega(\kappa)}$ . Moreover, the pad will be information theoretically secure.*

PROOF: Follows from Claim 15.49, Claim 15.50, Claim 15.51, Claim 15.52, Claim 15.53 and Claim 15.54.  $\square$

## 15.7 Communication Optimal SRMT Protocol in Directed Network

We now present an SRMT protocol called  $u$ -Optimal-SRMT-Static-Byzantine-Directed which sends a message  $m^{\mathbf{S}}$  containing  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements with very high probability, where  $\ell = (t_b - \frac{u}{2} + 1)n^2 = \Theta(n^3)$ . The total communication complexity of the protocol is  $\mathcal{O}(n^3)$  field elements and the protocol terminates in  $\mathcal{O}(u)$  phases. Thus the protocol achieves reliability with constant factor overhead.

The idea behind the protocol is to create a win-win situation with the adversary as follows: if the adversary corrupts at most  $t_b - \frac{u}{2}$  wires in the top band, then majority of the wires in the top band will be honest and  $\mathbf{R}$  recovers the message from the information which it receives from the honest wires in the top band. On the other hand, if more than  $t_b - \frac{u}{2}$  wires are corrupted in the top band, then majority wires in the bottom band will be honest and so both  $\mathbf{S}$  and  $\mathbf{R}$  comes to know about the identity of corrupted wires in the top band by using the honest wires in the bottom band. After knowing the identity of corrupted wires in the top band,  $\mathbf{S}$  re-sends  $m^{\mathbf{S}}$  so that  $\mathbf{R}$  can recover it correctly.

### 15.7.1 Pre-Processing Step of the Protocol

As a part of pre-processing step of protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed,  $\mathbf{S}$  and  $\mathbf{R}$  securely establishes  $\Theta(n^2)$  random non-zero elements from  $\mathbb{F}$  with each other in

advance with very high probability by executing the six phase protocol **6-Pad**. This will require a communication complexity of  $\mathcal{O}(n^3)$  field elements. Let the set of elements in the established pad be denoted by  $\mathcal{K}$ , which we call as **global key set**. The elements in  $\mathcal{K}$  will be used by **S** and **R** as authentication and hash keys to reliably exchange the outcome of certain steps during the execution of the protocol *u-Optimal-SRMT-Static-Byzantine-Directed*. Note that the elements in  $\mathcal{K}$  need not be distinct, but they are randomly selected from  $\mathbb{F}$ . We assume that initially all the elements in  $\mathcal{K}$  are marked as "unused". Each time **S** (**R**) needs a key(s) for hashing or authentication, then the first "unused" element(s) from  $\mathcal{K}$  is/are selected as key(s). In order to do the verification, **R** (**S**) also uses the same element(s) from  $\mathcal{K}$  as keys. Once the verification is done, the element(s) is/are marked as "used". Thus we can view  $\mathcal{K}$  as a global set, which is parallelly used and updated by both **S** and **R**. We now describe protocol *u-Optimal-SRMT-Static-Byzantine-Directed* phase by phase. We begin with the description of first two phases, which is given in the next section.

### 15.7.2 First Two Phases of the Protocol

The first two phases of protocol *u-Optimal-SRMT-Static-Byzantine-Directed* are given in Fig. 15.10.

We now prove the properties of first two phases of protocol *u-Optimal-SRMT-Static-Byzantine-Directed*.

**Claim 15.56** *Let  $f_i$  be a corrupted wire which has delivered  $F_i^{\mathbf{R}} \neq F_i^{\mathbf{S}}$  to **R** and let  $f_j$  be an honest wire. Then except with error probability  $2^{-\Omega(\kappa)}$ , the arc  $(f_i, f_j) \in \mathcal{E}^{\mathbf{R}}$ .*

PROOF: Since  $f_j$  is honest, it correctly and securely delivers  $\alpha_j^{\mathbf{R}} = \alpha_j^{\mathbf{S}}$  and  $v_{ij}^{\mathbf{R}} = v_{ij}^{\mathbf{S}} = \text{hash}(\alpha_j^{\mathbf{S}}; F_i^{\mathbf{S}})$  to **R**. If  $F_i^{\mathbf{R}} \neq F_i^{\mathbf{S}}$ , then in order that  $(f_i, f_j) \notin \mathcal{E}^{\mathbf{R}}$ ,  $\text{hash}(\alpha_j^{\mathbf{S}}; F_i^{\mathbf{S}}) = \text{hash}(\alpha_j^{\mathbf{S}}; F_i^{\mathbf{R}})$  should hold, even if  $F_i^{\mathbf{R}} \neq F_i^{\mathbf{S}}$ . But from the property of hashing, this can happen with probability at most  $\frac{n^2-1}{|\mathbb{F}|} \approx 2^{-\Omega(\kappa)}$ , which is negligible.  $\square$

**Claim 15.57** *Let  $f_i$  be a corrupted wire which has delivered  $F_i^{\mathbf{R}} \neq F_i^{\mathbf{S}}$  to **R**. Then except with probability  $2^{-\Omega(\kappa)}$ , there will exist at least one arc  $(f_i, f_j) \in \mathcal{E}^{\mathbf{R}}$ , such that wire  $f_j$  is honest.*

PROOF: The proof follows from the previous claim and the fact there exists at least one honest wire in the top band.  $\square$

**Claim 15.58** *In protocol *u-Optimal-SRMT-Static-Byzantine-Directed*, **S** communicates  $\mathcal{O}(n^3)$  field elements during **Phase I**, while **R** communicates  $\mathcal{O}(n^3)$  field elements during **Phase II**.*

PROOF: During **Phase I**, **S** communicates  $\mathcal{O}(n^2)$  field elements over each wire. This incurs a total communication complexity of  $\mathcal{O}(n^3)$  field elements. During **Phase II**, **R** sends the conflict list over the entire bottom band. In the worst case, the size of the conflict list may be  $\mathcal{O}(n^2)$  field elements. So sending it over entire bottom band requires a communication cost of  $\mathcal{O}(n^2u) = \mathcal{O}(n^3)$  field elements, as  $u = \mathcal{O}(n)$ .  $\square$

**Claim 15.59** *In protocol *u-Optimal-SRMT-Static-Byzantine-Directed*, if there exists at most  $t_b - \frac{u}{2}$  corrupted wires in the top band, then each wire  $f_i \in \mathcal{P}^{\mathbf{R}}$  will deliver correct  $F_i^{\mathbf{R}} = F_i^{\mathbf{S}}$  to **R**, except with probability  $2^{-\Omega(\kappa)}$ .*

Figure 15.10: First Two Phases of Protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed:  
 $|m^{\mathbf{S}}| = (t_b - \frac{u}{2} + 1)n^2$

**Phase I: S to R:** **S** does the following computation and communication:

1. **S** sets  $ROW = (t_b - \frac{u}{2} + 1)$ ,  $COL = n^2$ ,  $N = n$  and  $B_{init} = m^{\mathbf{S}}$  and executes Extrapolation Technique to construct an  $n \times n^2$  matrix  $B_{ext} = B^{\mathbf{S}}$  from  $B_{init}$ .
2. For  $i = 1, \dots, n$ , **S** selects a random, non-zero hash key  $\alpha_i^{\mathbf{S}}$  corresponding to wire  $f_i$ .
3. For  $i = 1, \dots, n$ , **S** sends the following to **R** over wire  $f_i$ :
  - (a) The  $n^2$ -tuple  $F_i^{\mathbf{S}}$ , which is the  $i^{th}$  row of  $B^{\mathbf{S}}$ .
  - (b) The hash key  $\alpha_i^{\mathbf{S}}$  and
  - (c) The hash values  $v_{ji}^{\mathbf{S}}$ , where  $v_{ji}^{\mathbf{S}} = hash(\alpha_i^{\mathbf{S}}; F_j^{\mathbf{S}})$ , for  $j = 1, \dots, n$ .

**Phase II: R to S:** **R** does the following computation and communication:

1. For  $i = 1, \dots, n$ , let **R** receive the following from **S** over wire  $f_i$ :
  - (a) The  $n^2$ -tuple  $F_i^{\mathbf{R}}$ ;
  - (b) The hash key  $\alpha_i^{\mathbf{R}}$  and
  - (c) The hash values  $v_{ji}^{\mathbf{R}}$ , for  $j = 1, \dots, n$ .
2. For  $i = 1, \dots, n$ , **R** computes  $Support_i = |\{j : v_{ij}^{\mathbf{R}} = hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{R}})\}|$ .
3. Let  $\mathcal{P}^{\mathbf{R}}$  denote the set of wires  $f_i$ , such that  $Support_i \geq (t_b - \frac{u}{2} + 1)$ .
4. **R** constructs a directed graph  $\mathcal{G}^{\mathbf{R}} = (\mathcal{V}^{\mathbf{R}}, \mathcal{E}^{\mathbf{R}})$ , called *conflict graph*, where  $\mathcal{V}^{\mathbf{R}} = \{f_1, f_2, \dots, f_n\}$  and arc  $(f_i, f_j) \in \mathcal{E}^{\mathbf{R}}$  if  $v_{ij}^{\mathbf{R}} \neq hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{R}})$ .
5. Corresponding to graph  $\mathcal{G}^{\mathbf{R}}$ , **R** constructs a conflict list  $\mathcal{Y}^{\mathbf{R}}$  of five tuples where for each arc  $(f_i, f_j) \in \mathcal{E}^{\mathbf{R}}$ , there exists a five tuple  $(f_i, f_j, \alpha_j^{\mathbf{R}}, hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{R}}), v_{ij}^{\mathbf{R}})$  in  $\mathcal{Y}^{\mathbf{R}}$ .
6. **R** sends  $\mathcal{Y}^{\mathbf{R}}$  to **S** through all the wires in the bottom band.

PROOF: Suppose there exists  $t_b - \frac{u}{2}$  corrupted wires in the top band. Let  $f_i \in \mathcal{P}^{\mathbf{R}}$ . If  $f_i$  is honest then it implies that  $F_i^{\mathbf{R}} = F_i^{\mathbf{S}}$ . On the other hand let  $f_i$  be a corrupted wire, who has delivered  $F_i^{\mathbf{R}} \neq F_i^{\mathbf{S}}$ . Since  $f_i \in \mathcal{P}^{\mathbf{R}}$ , it implies that  $|Support_i| \geq (t_b - \frac{u}{2} + 1)$ , which further implies that there exists at least one honest wire, say  $f_j$ , such that  $j \in Support_i$ . This further implies that  $v_{ij}^{\mathbf{R}} = hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{R}})$ . However, since  $f_j$  is an honest wire, it implies that  $\alpha_j^{\mathbf{R}} = \alpha_j^{\mathbf{S}}$  and  $v_{ij}^{\mathbf{R}} = v_{ij}^{\mathbf{S}}$ , where  $v_{ij}^{\mathbf{S}} = hash(\alpha_j^{\mathbf{S}}; F_i^{\mathbf{S}})$ . Since the adversary does not know  $\alpha_j^{\mathbf{S}}$ , it follows from the properties of hashing that the adversary can ensure that  $hash(\alpha_j^{\mathbf{S}}; F_i^{\mathbf{S}}) = hash(\alpha_j^{\mathbf{S}}; F_i^{\mathbf{R}})$ , even if  $F_i^{\mathbf{R}} = F_i^{\mathbf{S}}$ , except with probability  $\frac{n^2-1}{|\mathbb{F}|} \approx 2^{-\Omega(\kappa)}$ . Thus except with probability  $2^{-\Omega(\kappa)}$ ,  $f_i \notin \mathcal{P}^{\mathbf{R}}$ .  $\square$

**Lemma 15.60** *In protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed, if there exists at most  $t_b - \frac{u}{2}$  corrupted wires in the top band, then except with error probability  $2^{-\Omega(\kappa)}$ ,  $\mathbf{R}$  can correctly recover  $m^{\mathbf{R}} = m^{\mathbf{S}}$  by using the  $F_i^{\mathbf{R}}$ 's delivered by  $f_i$ 's in  $\mathcal{P}^{\mathbf{R}}$ .*

PROOF: If there exists at most  $t_b - \frac{u}{2}$  corrupted wires in the top band then from the proof of Claim 15.59, each  $f_i \in \mathcal{P}^{\mathbf{R}}$  has delivered correct  $F_i^{\mathbf{R}} = F_i^{\mathbf{S}}$ , except with error probability  $2^{-\Omega(\kappa)}$ . Moreover, there will be at least  $t_b - \frac{u}{2} + 1$  wires in  $\mathcal{P}^{\mathbf{R}}$ , as all honest wires in the top band will be present in  $\mathcal{P}^{\mathbf{R}}$ . Since  $ROW = (t_b - \frac{u}{2} + 1)$ , from the properties of Extrapolation Technique (see Lemma 8.10), by using the  $n^2$ -tuples delivered by the wires in  $\mathcal{P}^{\mathbf{R}}$ ,  $\mathbf{R}$  can reconstruct  $B^{\mathbf{S}}$  and hence  $B_{init}$ . Now the elements of  $B_{init}$  are nothing but the elements of  $m^{\mathbf{S}}$ .  $\square$

Lemma 15.60 shows that if somehow  $\mathbf{R}$  can find out whether there are at most  $t_b - \frac{u}{2}$  corrupted wires in the top band, then  $\mathbf{R}$  can recover  $m^{\mathbf{S}}$  by using the  $n^2$ -tuples delivered by the wires in  $\mathcal{P}^{\mathbf{R}}$ . In order to find the status of the top band,  $\mathbf{R}$  constructs the conflict list and sends it to  $\mathbf{S}$ . We now proceed to the description of **Phase III** of protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed, which is given in the next section.

### 15.7.3 Phase III and Phase IV of the Protocol

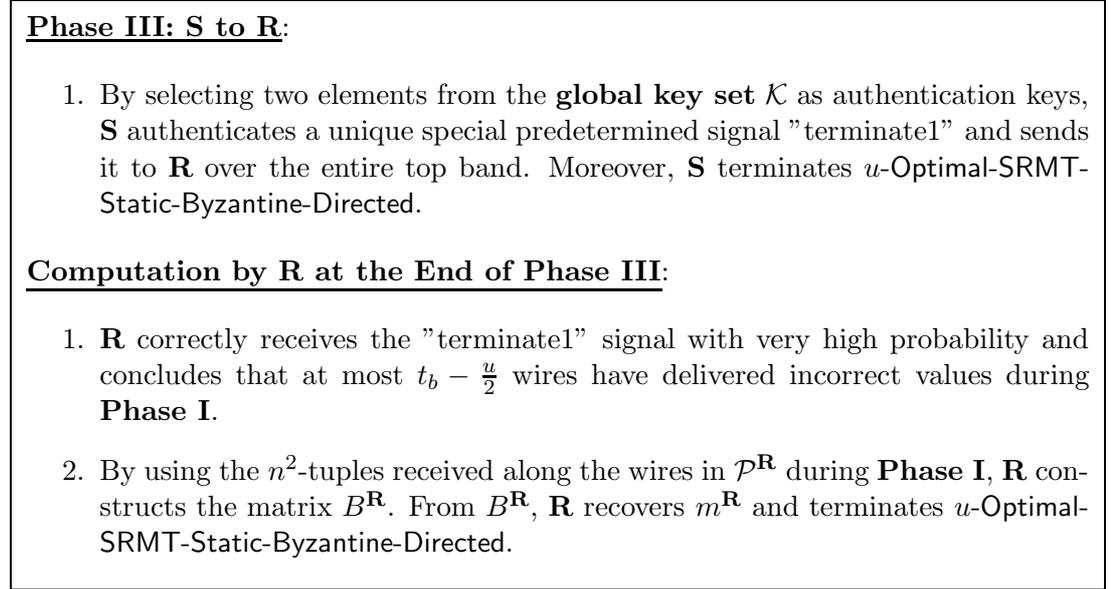
$\mathbf{S}$  waits for a conflict list, which is received identically through at least  $\frac{u}{2} + 1$  wires. If  $\mathbf{S}$  does not receive any conflict list identically through at least  $\frac{u}{2} + 1$  wires, then  $\mathbf{S}$  concludes that at least  $\frac{u}{2} + 1$  wires are corrupted in the bottom band. This further implies that at most  $t_b - \frac{u}{2} - 1$  wires are corrupted in the top band. In this case, the protocol proceeds as shown in Fig. 15.11.

The correctness of the protocol in this execution sequence is proved in Lemma 15.61.

**Lemma 15.61** *If  $\mathbf{S}$  does not receive the same conflict list through at least  $\frac{u}{2} + 1$  wires in the bottom band then except with error probability  $2^{-\Omega(\kappa)}$ ,  $\mathbf{R}$  correctly recovers  $m^{\mathbf{S}}$  from the  $n^2$ -tuples delivered by the wires in  $\mathcal{P}^{\mathbf{R}}$ . Moreover, in this case,  $u$ -Optimal-SRMT-Static-Byzantine-Directed terminates in three phases. Furthermore,  $\mathbf{S}$  will communicate  $\mathcal{O}(n)$  field elements during **Phase III**.*

PROOF: If  $\mathbf{S}$  does not receive the same conflict list through at least  $\frac{u}{2} + 1$  wires in the bottom band, then it implies that at least  $\frac{u}{2} + 1$  wires in the bottom band are corrupted which further implies that at most  $t_b - \frac{u}{2} - 1$  wires in the top band are corrupted. This further implies that each wire  $f_i \in \mathcal{P}^{\mathbf{R}}$  has delivered correct  $F_i^{\mathbf{R}} = F_i^{\mathbf{S}}$  to  $\mathbf{R}$  during **Phase I**, except with probability  $2^{-\Omega(\kappa)}$  (see Claim 15.59). Moreover, from the proof of Lemma of 15.60,  $\mathbf{R}$  can correctly recover  $m^{\mathbf{R}} = m^{\mathbf{S}}$  by using the  $F_i^{\mathbf{R}}$ 's delivered by  $f_i$ 's in  $\mathcal{P}^{\mathbf{R}}$ , except with error probability  $2^{-\Omega(\kappa)}$ .

Figure 15.11: Execution of  $u$ -Optimal-SRMT-Static-Byzantine-Directed If **S** Does Not Receive  $\frac{u}{2} + 1$  Identical Conflict Lists Through the Bottom Band



Since **S** authenticates the "terminate1" signal by using the keys from global key set  $\mathcal{K}$ , the keys will be unknown to the adversary. So except with error probability  $2^{-\Omega(\kappa)}$ , **R** will receive the "terminate1" signal and will conclude that at most  $t_b - \frac{u}{2} - 1$  wires in the top band are corrupted. Now as explained above, **R** will correctly recover  $m^{\mathbf{S}}$  from the  $n^2$ -tuples delivered by the wires in  $\mathcal{P}^{\mathbf{R}}$ .

It is easy to see that in this case, protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed terminates in three phases. Since **S** only sends the authentication of "terminate1" signal over each wire during **Phase III**, it will require a communication cost of  $\mathcal{O}(n)$  field elements.  $\square$

If **S** receives a unique conflict list, say  $\mathcal{Y}^{\mathbf{S}}$ , through at least  $\frac{u}{2} + 1$  wires in the bottom band then **S** cannot conclude anything about the status of the top band and bottom band. That is, **S** cannot determine whether the received conflict list is a genuine conflict list or not. **S** considers the received conflict list as genuine and from it, after doing local comparison, **S** tries to find the number of corrupted wires, which delivered incorrect  $F_i^{\mathbf{S}}$ 's to **R** during **Phase I**. **S** saves the identity of such wires in a list  $L_{fault}^{\mathbf{S}}$ . The steps performed by **S** to compute  $L_{fault}^{\mathbf{S}}$  from  $\mathcal{Y}^{\mathbf{S}}$  is shown in Fig. 15.12.

Before proceeding further, we make the following claims:

**Claim 15.62** *Let majority of the wires in the bottom band are honest and let  $f_i$  be a corrupted wire in the top band who has delivered  $F_i^{\mathbf{R}} \neq F_i^{\mathbf{S}}$  to **R** during **Phase I**. Then except with error probability  $2^{-\Omega(\kappa)}$ , **S** will include  $f_i$  in  $L_{fault}^{\mathbf{S}}$  at the end of **Phase II**.*

PROOF: Let  $f_i$  be a corrupted wire in the top band who has delivered  $F_i^{\mathbf{R}} \neq F_i^{\mathbf{S}}$  to **R** during **Phase I**. Then from Claim 15.57, except with error probability  $2^{-\Omega(\kappa)}$ , there will exist at least one arc  $(f_i, f_j)$  in the conflict graph, such that  $f_j$  is an honest wire. This further implies that the five tuple  $(f_i, f_j, \alpha_j^{\mathbf{R}}, hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{R}}), v_{ij}^{\mathbf{R}})$  will be present in the conflict list  $\mathcal{Y}^{\mathbf{R}}$ , except with error probability  $2^{-\Omega(\kappa)}$ , such that  $\alpha_j^{\mathbf{R}} = \alpha_j^{\mathbf{S}}$ ,  $v_{ij}^{\mathbf{R}} = v_{ij}^{\mathbf{S}}$ , but  $hash(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{R}}) \neq v_{ij}^{\mathbf{R}}$ . Now if the majority of the wires in the bottom

Figure 15.12: Local Computation by **S** at the End of Phase II if **S** Receives  $\frac{u}{2} + 1$  Identical Conflict Lists Through the Bottom Band

**Local Computation by **S** at the End of Phase II:**

1. Let **S** receive the conflict list  $\mathcal{Y}^{\mathbf{S}}$  through at least  $\frac{u}{2} + 1$  wires in the bottom band, where  $\mathcal{Y}^{\mathbf{S}}$  is a collection of five-tuple of the form  $(f_i, f_j, \alpha_j^{\mathbf{S}}, \gamma_{ij}^{\mathbf{S}}, v_{ij}^{\mathbf{S}})$ .
2. **S** creates a list  $L_{fault}^{\mathbf{S}}$ , which is initialized to  $\emptyset$ .
3. For each five-tuple  $(f_i, f_j, \alpha_j^{\mathbf{S}}, \gamma_{ij}^{\mathbf{S}}, v_{ij}^{\mathbf{S}})$  in  $\mathcal{Y}^{\mathbf{S}}$ , **S** does the following computation:
  - (a) **S** checks  $\alpha_j^{\mathbf{S}} \stackrel{?}{=} \alpha_j^{\mathbf{S}}$  and  $v_{ij}^{\mathbf{S}} \stackrel{?}{=} v_{ij}^{\mathbf{S}}$ .
  - (b) If any of the above test fails then **S** concludes that wire  $f_j$  has delivered incorrect values to **R** during **Phase I** and adds  $f_j$  to a list  $L_{fault}^{\mathbf{S}}$ .
  - (c) If  $f_j$  is not added to  $L_{fault}^{\mathbf{S}}$  then **S** checks  $\gamma_{ij}^{\mathbf{S}} \stackrel{?}{=} \text{hash}(\alpha_j^{\mathbf{S}}; F_i^{\mathbf{S}})$ .
  - (d) If the above test fails then **S** concludes that wire  $f_i$  has delivered incorrect  $F_i^{\mathbf{R}} \neq F_i^{\mathbf{S}}$  to **R** during **Phase I** and adds  $f_i$  to  $L_{fault}^{\mathbf{S}}$ .

band are honest then **S** will correctly receive  $\mathcal{Y}^{\mathbf{S}} = \mathcal{Y}^{\mathbf{R}}$  through the majority wires in the bottom band. This implies that **S** will correctly receive  $(f_i, f_j, \alpha_j^{\mathbf{S}}, \gamma_{ij}^{\mathbf{S}}, v_{ij}^{\mathbf{S}}) = (f_i, f_j, \alpha_j^{\mathbf{R}}, \text{hash}(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{R}}), v_{ij}^{\mathbf{R}})$ . So after doing the local comparison, **S** will find that  $\alpha_j^{\mathbf{S}} = \alpha_j^{\mathbf{R}}$  and  $v_{ij}^{\mathbf{S}} = v_{ij}^{\mathbf{R}}$  and hence  $f_j$  will not be added in  $L_{fault}^{\mathbf{S}}$ . But **S** will find that  $\gamma_{ij}^{\mathbf{S}} \neq \text{hash}(\alpha_j^{\mathbf{S}}; F_i^{\mathbf{S}})$ . So **S** will add wire  $f_i$  in  $L_{fault}^{\mathbf{S}}$ .  $\square$

**Claim 15.63** *Let majority of the wires in the bottom band are honest. Then **S** will not include any honest  $f_i$  in  $L_{fault}^{\mathbf{S}}$  at the end of **Phase II**.*

PROOF: First of all notice that if at all a five tuple  $(f_i, f_j, \alpha_j^{\mathbf{R}}, \text{hash}(\alpha_j^{\mathbf{R}}; F_i^{\mathbf{R}}), v_{ij}^{\mathbf{R}})$  is present in the conflict list  $\mathcal{Y}^{\mathbf{R}}$ , then at least one of the wires  $f_i$  or  $f_j$  is corrupted. This is because no two honest wires will conflict each other. Now suppose  $f_i$  is the corrupted wire. If majority of the wires in the bottom band are honest then **S** will correctly receive  $\mathcal{Y}^{\mathbf{S}} = \mathcal{Y}^{\mathbf{R}}$  through the majority wires in the bottom band. Now as shown in the previous claim, after doing local comparison, **S** will find out that wire  $f_i$  is corrupted and includes only  $f_i$  in  $L_{fault}^{\mathbf{S}}$ .  $\square$

After finding  $L_{fault}^{\mathbf{S}}$ , **S** proceeds as follows: If  $L_{fault}^{\mathbf{S}}$  contains at most  $t_b - \frac{u}{2}$  wires then **S** can conclude that **R** has received sufficient information during **Phase I** to recover  $m^{\mathbf{S}}$ . This is because if at all  $\mathcal{Y}^{\mathbf{S}}$  is a valid conflict list, which was indeed sent by **R** then the wires in  $L_{fault}^{\mathbf{S}}$  are indeed genuinely corrupted, who delivered incorrect  $F_i^{\mathbf{S}}$ 's to **R** during **Phase I** (see Claim 15.62) and there are at most  $t_b - \frac{u}{2}$  such  $F_i^{\mathbf{S}}$ 's. This implies that the remaining  $t_b - \frac{u}{2} + 1$  wires have delivered correct  $F_i^{\mathbf{S}}$ 's, which are sufficient to reconstruct the matrix  $B^{\mathbf{S}}$  and hence the message  $m^{\mathbf{S}}$ .

On the other hand, if  $\mathcal{Y}^{\mathbf{S}}$  is not a valid conflict list, then it implies that majority of wires in the bottom band are corrupted, which further implies that majority of wires in the top band are honest. So the wires in  $\mathcal{P}^{\mathbf{R}}$  have delivered correct  $F_i^{\mathbf{S}}$ 's during **Phase I** with very high probability.

Thus if  $\mathbf{S}$  finds that  $L_{fault}^{\mathbf{S}}$  contains at most  $t_b - \frac{u}{2}$  wires then the main goal of  $\mathbf{S}$  will be to somehow reliably send back the received conflict list  $\mathcal{Y}^{\mathbf{S}}$  and the corresponding list  $L_{fault}^{\mathbf{S}}$ , so that  $\mathbf{R}$  can find out whether  $\mathbf{S}$  has correctly received the original conflict list  $\mathcal{Y}^{\mathbf{R}}$  through the majority wires in the bottom band. This is what  $\mathbf{S}$  does during **Phase III**, as shown in Fig. 15.13. Notice that Fig. 15.13 represents the execution sequence when  $\mathbf{S}$  finds that there are at most  $t_b - \frac{u}{2}$  wires in the list  $L_{fault}^{\mathbf{S}}$ . The execution sequence when  $L_{fault}^{\mathbf{S}}$  contains more than  $t_b - \frac{u}{2}$  wires will be discussed afterwards.

We now prove the properties of protocol *u-Optimal-SRMT-Static-Byzantine-Directed*, if the protocol follows the steps given in Fig. 15.13.

**Lemma 15.64** *If  $\mathbf{S}$  receives the same conflict list through at least  $\frac{u}{2} + 1$  wires in the bottom band and if the size of resultant  $L_{fault}^{\mathbf{S}}$  is at most  $(t_b - \frac{u}{2})$  then except with error probability  $2^{-\Omega(\kappa)}$ ,  $\mathbf{R}$  will correctly output  $m^{\mathbf{R}} = m^{\mathbf{S}}$  at the end of **Phase III**. In this case, protocol *u-Optimal-SRMT-Static-Byzantine-Directed* will terminate in three phases and  $\mathbf{S}$  will communicate  $\mathcal{O}(n^3)$  field elements during **Phase III**.*

PROOF: Let  $\mathbf{S}$  receive the conflict graph  $\mathcal{Y}^{\mathbf{S}}$  through at least  $\frac{u}{2} + 1$  wires in the bottom band and let  $L_{fault}^{\mathbf{S}}$  be the corresponding list computed by  $\mathbf{S}$  from  $\mathcal{Y}^{\mathbf{S}}$ . Moreover, let  $|L_{fault}^{\mathbf{S}}| \leq (t_b - \frac{u}{2})$ . Now there are two possibilities:

1.  $\mathcal{Y}^{\mathbf{S}} = \mathcal{Y}^{\mathbf{R}}$ : In this case, except with probability  $2^{-\Omega(\kappa)}$ ,  $\mathbf{S}$  will find out all corrupted  $f_i$ 's, who have delivered  $F_i^{\mathbf{R}} \neq F_i^{\mathbf{S}}$  during **Phase I** and includes them in  $L_{fault}^{\mathbf{S}}$  (see Claim 15.62). Moreover, from Claim 15.63, only corrupted  $f_i$ 's will be included in  $L_{fault}^{\mathbf{S}}$ . Since  $|L_{fault}^{\mathbf{S}}| \leq (t_b - \frac{u}{2})$ , it implies that at least  $(t_b - \frac{u}{2}) + 1$   $f_i$ 's delivered correct  $F_i^{\mathbf{S}}$ 's during **Phase I**. Now during **Phase III**,  $\mathbf{S}$  sends the received  $\mathcal{Y}^{\mathbf{S}}$ , corresponding  $L_{fault}^{\mathbf{S}}$ , authentication of  $\mathcal{Y}^{\mathbf{S}}$  and the authentication of  $L_{fault}^{\mathbf{S}}$  through the entire top band. Notice that the authentication keys used to authenticate  $\mathcal{Y}^{\mathbf{S}}$  and  $L_{fault}^{\mathbf{S}}$  belong to  $\mathcal{K}$  and so adversary does not have any information about them. Moreover, there exists at least one honest wire in the top band. So from the properties of *URauth*,  $\mathbf{R}$  will correctly receive  $L_{fault}^{\mathbf{S}}$  with very high probability. By neglecting the wires in  $L_{fault}^{\mathbf{S}}$ ,  $\mathbf{R}$  will be left with at least  $(t_b - \frac{u}{2} + 1)$  wires in the top band and each of them has delivered correct  $F_i^{\mathbf{S}}$  with very high probability. Now it is easy to see that using these  $F_i^{\mathbf{S}}$ 's,  $\mathbf{R}$  will correctly recover  $B^{\mathbf{S}}$  and hence  $m^{\mathbf{S}}$ . It is easy to see that in this case protocol *u-Optimal-SRMT-Static-Byzantine-Directed* terminates at the end of **Phase III**.
2.  $\mathcal{Y}^{\mathbf{S}} \neq \mathcal{Y}^{\mathbf{R}}$ : This implies that at least  $\frac{u}{2} + 1$  wires in the bottom band are corrupted, which further implies that at most  $t_b - \frac{u}{2} - 1$  wires are corrupted in the top band. So from Claim 15.59, all wires in  $\mathcal{P}^{\mathbf{R}}$  have delivered correct  $F_i^{\mathbf{R}}$ 's during **Phase I** with very high probability. Moreover, if somehow  $\mathbf{R}$  comes to know that there are at most  $t_b - \frac{u}{2} - 1$  corrupted wires in the top band, then from the proof of Lemma 15.60,  $\mathbf{R}$  can recover  $m^{\mathbf{S}}$  correctly with very high probability by using the  $F_i^{\mathbf{R}}$ 's delivered by the wires in  $\mathcal{P}^{\mathbf{R}}$ . We now show that at the end of **Phase III**, with very high probability,  $\mathbf{R}$  will come to know that at most  $t_b - \frac{u}{2} - 1$  wires are corrupted in the top band.

Note that  $\mathbf{S}$  computes  $L_{fault}^{\mathbf{S}}$  by doing local comparisons on the values in  $\mathcal{Y}^{\mathbf{S}}$ . Also  $\mathcal{Y}_{auth}^{\mathbf{S}}$  and  $L_{fault_{auth}}^{\mathbf{S}}$  are obtained by applying *URauth* function to the elements of  $\mathcal{Y}^{\mathbf{S}}$  and  $L_{fault}^{\mathbf{S}}$  respectively, where the keys for authentication are selected

from secret, global key set  $\mathcal{K}$ . So adversary will have no information about the authentication keys used for authenticating  $\mathcal{Y}^{\mathbf{S}}$  and  $L_{fault}^{\mathbf{S}}$ . During **Phase III**,  $\mathbf{S}$  sends  $(\mathcal{Y}^{\mathbf{S}}, L_{fault}^{\mathbf{S}}, \mathcal{Y}_{auth}^{\mathbf{S}}, L_{fault_{auth}}^{\mathbf{S}})$  to  $\mathbf{R}$  through the top band.

Now suppose that some wire  $f_i$  delivers  $(\mathcal{Y}_i^{\mathbf{R}}, L_{fault_i}^{\mathbf{R}}, \mathcal{Y}_{i,auth}^{\mathbf{R}}, L_{fault_{i,auth}}^{\mathbf{R}})$ . If  $f_i$  is honest, then  $\mathcal{Y}_i^{\mathbf{R}} = \mathcal{Y}^{\mathbf{S}} \neq \mathcal{Y}^{\mathbf{R}}$ . So  $\mathbf{R}$  will neglect wire  $f_i$ . On the other hand if  $f_i$  is corrupted, then the adversary can ensure that  $\mathcal{Y}_i^{\mathbf{R}} \neq \mathcal{Y}^{\mathbf{S}}$  but  $\mathcal{Y}_i^{\mathbf{R}} = \mathcal{Y}^{\mathbf{R}}$ . However, from the properties of  $UR_{auth}$ , without knowing the authentication keys used by  $\mathbf{S}$  for authenticating  $\mathcal{Y}^{\mathbf{S}}$ , the adversary cannot produce the authentication of  $\mathcal{Y}_i^{\mathbf{R}} = \mathcal{Y}^{\mathbf{R}}$ , except with probability  $2^{-\Omega(\kappa)}$ . So except with error probability  $2^{-\Omega(\kappa)}$ , wire  $f_i$  will be caught and hence will be discarded by  $\mathbf{R}$ . This further implies that except with error probability all wires in top band will be discarded by  $\mathbf{R}$  and hence  $\mathbf{R}$  will conclude that  $\mathbf{S}$  has not received  $\mathcal{Y}^{\mathbf{R}}$  through at least  $\frac{u}{2} + 1$  wires in the bottom band. Now as explained above,  $\mathbf{R}$  can recover  $m^{\mathbf{S}}$  correctly with very high probability by using the  $F_i^{\mathbf{R}}$ 's delivered by the wires in  $\mathcal{P}^{\mathbf{R}}$ . It is easy to see that in this case protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed terminates at the end of **Phase III**.

Notice that if  $\mathbf{S}$  executes **Phase III** as given in Fig. 15.13, then  $\mathbf{S}$  sends  $(\mathcal{Y}^{\mathbf{S}}, L_{fault}^{\mathbf{S}}, \mathcal{Y}_{auth}^{\mathbf{S}}, L_{fault_{auth}}^{\mathbf{S}})$  through entire top band. It is easy to see that this requires a communication complexity of  $\mathcal{O}(n^3)$  field elements, as  $|\mathcal{Y}^{\mathbf{S}}| = \mathcal{O}(n^2)$ .  $\square$

Now we draw our attention to the execution of  $u$ -Optimal-SRMT-Static-Byzantine-Directed when  $\mathbf{S}$  receive  $\mathcal{Y}^{\mathbf{S}}$  through  $\frac{u}{2} + 1$  wires in the bottom band and the resultant  $L_{fault}^{\mathbf{S}}$  has more than  $(t_b - \frac{u}{2})$  wires. In this case,  $\mathbf{S}$  cannot say anything about the status of top band. To find whether indeed  $\mathcal{Y}^{\mathbf{S}} = \mathcal{Y}^{\mathbf{R}}$ ,  $\mathbf{S}$  authenticates  $\mathcal{Y}^{\mathbf{S}}$  and  $L_{fault}^{\mathbf{S}}$ , sends it to  $\mathbf{R}$  and wait for the feedback. If indeed  $\mathcal{Y}^{\mathbf{S}} = \mathcal{Y}^{\mathbf{R}}$  then the wires in  $L_{fault}^{\mathbf{S}}$  are indeed corrupted and so  $\mathbf{S}$  and  $\mathbf{R}$  ignores them and further continue with the protocol. The steps for further computation and communication will be discusses later. However, if  $\mathcal{Y}^{\mathbf{S}} \neq \mathcal{Y}^{\mathbf{R}}$  then with very high probability,  $\mathbf{R}$  will detect this. In this case,  $\mathbf{R}$  can easily recover  $m^{\mathbf{S}}$  by using the  $F_i^{\mathbf{R}}$ 's delivered by the wires in  $\mathcal{P}^{\mathbf{R}}$  during **Phase I**. So  $\mathbf{R}$  recovers  $m^{\mathbf{S}}$  and asks  $\mathbf{S}$  to terminate the protocol. The formal details are presented in Fig. 15.14.

We now prove the properties of protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed, if the protocol follows the steps given in Fig. 15.14.

**Lemma 15.65** *If  $\mathbf{S}$  receives the same conflict list  $\mathcal{Y}^{\mathbf{S}}$  through at least  $\frac{u}{2} + 1$  wires in the bottom band and if the size of resultant  $L_{fault}^{\mathbf{S}}$  is more than  $(t_b - \frac{u}{2} + 1)$  then:*

1. *If  $\mathcal{Y}^{\mathbf{S}} = \mathcal{Y}^{\mathbf{R}}$  then both  $\mathbf{R}$  and  $\mathbf{S}$  will come to know about this at the end of **Phase III** and **Phase IV** respectively. Moreover, both  $\mathbf{S}$  and  $\mathbf{R}$  will come to know the identity of  $|L_{fault}^{\mathbf{S}}| \geq (t_b - \frac{u}{2} + 1)$  corrupted wires in the top band. Furthermore, both  $\mathbf{S}$  and  $\mathbf{R}$  will know that the majority of the wires in the bottom band are honest.*
2. *If  $\mathcal{Y}^{\mathbf{S}} \neq \mathcal{Y}^{\mathbf{R}}$  then except with probability  $2^{-\Omega(\kappa)}$ ,  $\mathbf{R}$  will come to know this at the end of **Phase III**. Moreover,  $\mathbf{R}$  will recover  $m^{\mathbf{R}}$  at the end of **Phase III** by using the  $F_i^{\mathbf{R}}$ 's delivered by the wires in  $\mathcal{P}^{\mathbf{R}}$ . Furthermore,  $\mathbf{S}$  will terminate the protocol at the end of **Phase IV**.*
3. *During **Phase III**,  $\mathbf{S}$  will communicate  $\mathcal{O}(n^3)$  field elements while during **Phase IV**,  $\mathbf{R}$  will communicate  $\mathcal{O}(u)$  field elements.*

PROOF: Let  $\mathbf{S}$  receive the same conflict list  $\mathcal{Y}^{\mathbf{S}}$  through at least  $\frac{u}{2} + 1$  wires in the bottom band at the end of **Phase II**. Moreover, let the resultant  $L_{fault}^{\mathbf{S}}$  computed from  $\mathcal{Y}^{\mathbf{S}}$  be of size more than  $(t_b - \frac{u}{2} + 1)$ . Now there are two possible cases:

1.  $\mathcal{Y}^{\mathbf{S}} = \mathcal{Y}^{\mathbf{R}}$ : This case is similar to the case  $\mathcal{Y}^{\mathbf{S}} = \mathcal{Y}^{\mathbf{R}}$  in the proof of Lemma 15.64. Specifically, at the end of **Phase III**,  $\mathbf{R}$  will conclude that  $\mathbf{S}$  has correctly received  $\mathcal{Y}^{\mathbf{R}}$  over the majority wires in the bottom band during **Phase II**. Moreover, except with error probability  $2^{-\Omega(\kappa)}$ ,  $\mathbf{R}$  will correctly receive  $L_{fault}^{\mathbf{S}}$ . Furthermore, each wire in  $L_{fault}^{\mathbf{S}}$  will be indeed corrupted. So  $\mathbf{R}$  will remove the wires in  $L_{fault}^{\mathbf{S}}$  from his consideration for all future computation and communication. Since  $|L_{fault}^{\mathbf{S}}| \geq (t_b - \frac{u}{2} + 1)$ , this implies that in the bottom band, there will be at most  $\frac{u}{2} - 1$  corrupted wires and hence  $\mathbf{R}$  concludes that the majority of the wires in the bottom band are honest. Since in this case  $\mathbf{R}$  authenticates "continue" signal and sends to  $\mathbf{S}$ , at the end of **Phase IV**,  $\mathbf{S}$  will correctly receive the signal over at least  $\frac{u}{2} + 1$  wires in the bottom band and thus will conclude that  $\mathcal{Y}^{\mathbf{S}} = \mathcal{Y}^{\mathbf{R}}$ . So  $\mathbf{S}$  will also remove the wires in  $L_{fault}^{\mathbf{S}}$  from his consideration for all future computation and communication and will conclude that majority of the wires in the bottom band are honest. This proves the first part of the lemma.
2.  $\mathcal{Y}^{\mathbf{S}} \neq \mathcal{Y}^{\mathbf{R}}$ : This case is similar to the case  $\mathcal{Y}^{\mathbf{S}} \neq \mathcal{Y}^{\mathbf{R}}$  in the proof of Lemma 15.64. Specifically, at the end of **Phase III**, except with probability  $2^{-\Omega(\kappa)}$ ,  $\mathbf{R}$  will conclude that  $\mathbf{S}$  has not correctly received  $\mathcal{Y}^{\mathbf{R}}$  over the majority wires in the bottom band during **Phase II**. This further implies that there are at most  $t_b - \frac{u}{2} - 1$  corrupted wires in the top band, which further implies that the  $F_i^{\mathbf{R}}$ 's delivered by the wires in  $\mathcal{P}^{\mathbf{R}}$  during **Phase I** are correct, except with error probability  $2^{-\Omega(\kappa)}$ . So using these  $F_i^{\mathbf{R}}$ 's,  $\mathbf{R}$  will correctly recover  $m^{\mathbf{S}}$ , except with error probability  $2^{-\Omega(\kappa)}$ . Moreover, in this case,  $\mathbf{R}$  asks  $\mathbf{S}$  to terminate the protocol by authenticating "terminate" signal and sending it through the bottom band. Since the keys used for authenticating "terminate" signal are selected from  $\mathcal{K}$ , they will be unknown to the adversary. So from the properties of  $URauth$ , the adversary cannot generate authentication of any signal, other than "terminate" and put into the bottom band. So if at all  $\mathbf{S}$  receives a valid authenticated signal from at least  $\frac{u}{2} + 1$  wires in the bottom band, it has to be "terminate" signal, except with error probability  $2^{-\Omega(\kappa)}$ . On receiving the signal,  $\mathbf{S}$  also comes to know that  $\mathcal{Y}^{\mathbf{S}} \neq \mathcal{Y}^{\mathbf{R}}$  and terminates the protocol. On the other hand if  $\mathbf{S}$  does not receive a valid authenticated signal from at least  $\frac{u}{2} + 1$  wires in the bottom band, then also  $\mathbf{S}$  comes to know that  $\mathcal{Y}^{\mathbf{S}} \neq \mathcal{Y}^{\mathbf{R}}$  and terminates the protocol. This proves the second part of the lemma.

The fact that  $\mathbf{S}$  communicates  $\mathcal{O}(n^3)$  field elements during **Phase III** follows from the proof of Lemma 15.65. During **Phase IV**,  $\mathbf{R}$  authenticates either "terminate" and "continue" signal and sends through the entire bottom band. This will require a communication complexity of  $\mathcal{O}(u)$  field elements. This proves the third part of the lemma.  $\square$

#### 15.7.4 Remaining Phases of the Protocol

Finally, we move towards the discussion of protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed when  $\mathbf{S}$  receives  $\mathcal{Y}^{\mathbf{S}}$  over majority of the wires in the bottom band during **Phase II**, such that the resultant  $L_{fault}^{\mathbf{S}}$  is of size more than  $t_b - \frac{u}{2} + 1$  and  $\mathbf{S}$  has received "continue" signal from  $\mathbf{R}$  through majority of the wires in the bottom band

during **Phase IV**. If this is the case, then from the proof of Lemma 15.65, except with error probability  $2^{-\Omega(\kappa)}$ , both **S** and **R** knows the identity of  $|L_{fault}^{\mathbf{S}}| \geq t_b - \frac{u}{2} + 1$  corrupted wires in the top band and removes them from their consideration. This implies that at most  $\frac{u}{2} - 1$  wires in the bottom band are corrupted and hence majority wires in the bottom band are honest. This further implies that the information received by **R** during **Phase I** is insufficient to reconstruct  $B^{\mathbf{S}}$ . This is because in this case more than  $t_b - \frac{u}{2} + 1$  wires have delivered incorrect  $F_i^{\mathbf{R}}$ 's during **Phase I**. But to reconstruct  $B^{\mathbf{S}}$ , **R** must have the knowledge of  $t_b - \frac{u}{2} + 1$  correct  $F_i^{\mathbf{R}}$ 's.

So **S** again starts re-sending  $m^{\mathbf{S}}$ . Notice that both **S** and **R** knows that at most  $\frac{u}{2} - 1$  wires in the top band are corrupted. Moreover, majority of the wires in the bottom band are honest. To resend  $m^{\mathbf{S}}$ , **S** considers only the first  $\frac{u}{2}$  wires in the top band, which are still in the consideration of **S** and **R**. *Without loss of generality, let these be the first  $\frac{u}{2}$  wires in the top band.* Both **S** and **R** knows that at least one wire among these  $\frac{u}{2}$  wires are honest. Notice that in order to re-send  $m^{\mathbf{S}}$ , **S** should not communicate more than  $\mathcal{O}(|m^{\mathbf{S}}|)$  field elements. This is because the overall goal of *u-Optimal-SRMT-Static-Byzantine-Directed* is to achieve reliability with constant factor overhead. It seems that there is no way **S** can re-send  $m^{\mathbf{S}}$  by communicating  $\mathcal{O}(|m^{\mathbf{S}}|)$  field elements, as he does not know the identity of the  $\frac{u}{2} - 1$  corrupted wires in the top band. However, the fact that at least one wire among the  $\frac{u}{2}$  wires in the top band is honest and majority wires in the bottom band are honest comes to our rescue !

**S** divides  $m^{\mathbf{S}}$  into  $\frac{u}{2}$  blocks and tries to sequentially send each block, one by one. Specifically, **S** considers the first block of  $m^{\mathbf{S}}$  and sends it only over wire  $f_1$  and waits for the authenticated feedback from **R**. If  $f_1$  has correctly delivered the block then **S** will come to know this and will continue with the second block. Otherwise with very high probability, **S** will come to know that  $f_1$  has not delivered the first block correctly. So **S** tells about this to **R** by sending an authenticated signal and then again send the first block of  $m^{\mathbf{S}}$  through the second wire. This process will continue till either all the blocks of  $m^{\mathbf{S}}$  have been delivered or **S** and **R** has tried the first  $\frac{u}{2} - 1$  wires. In the later case, both **S** and **R** knows that wire  $f_{\frac{u}{2}}$  is honest and so **S** sends all the remaining blocks of  $m^{\mathbf{S}}$  to **R** through wire  $f_{\frac{u}{2}}$ . It is easy to see that this entire process will take  $\mathcal{O}(u)$  phases and will require a communication complexity of  $\mathcal{O}(|m^{\mathbf{S}}|)$  field elements. The formal details of the protocol steps are given in Fig. 15.15.

We now prove the properties of protocol *u-Optimal-SRMT-Static-Byzantine-Directed* if the protocol follows the steps given in Fig. 15.15.

**Lemma 15.66** *Suppose **S** receives the same conflict list  $\mathcal{Y}^{\mathbf{S}}$  during **Phase II** through at least  $\frac{u}{2} + 1$  wires in the bottom band, such that the size of resultant  $L_{fault}^{\mathbf{S}}$  is more than  $(t_b - \frac{u}{2} + 1)$ . Moreover let **S** receive "continue" signal during **Phase IV** through majority wires in the bottom band. Then except with error probability  $2^{-\Omega(\kappa)}$ , **S** will be able to correctly re-send  $m^{\mathbf{S}}$  to **R** in  $\mathcal{O}(u)$  phases by following the steps given in Fig. 15.15. Moreover, this requires a communication complexity of  $\mathcal{O}(|m^{\mathbf{S}}|) = \mathcal{O}(n^3)$  field elements.*

PROOF: Suppose **S** has received the same conflict list  $\mathcal{Y}^{\mathbf{S}}$  during **Phase II** through at least  $\frac{u}{2} + 1$  wires in the bottom band, such that the size of resultant  $L_{fault}^{\mathbf{S}}$  is more than  $(t_b - \frac{u}{2} + 1)$ . Moreover let **S** has received "continue" signal during **Phase IV** through majority wires in the bottom band. Then from the proof of first part of Lemma 15.65, except with error probability  $2^{-\Omega(\kappa)}$ , at most  $\frac{u}{2} - 1$  wires in the top band and at most  $\frac{u}{2} - 1$  wires in the bottom band are corrupted. Moreover, **S** and **R** will know this. Without loss of generality, let the first  $\frac{u}{2}$  wires in the top band are still in the

consideration of  $\mathbf{S}$  and  $\mathbf{R}$ , after neglecting  $|L_{fault}^{\mathbf{S}}| \geq (t_b - \frac{u}{2} + 1)$  corrupted wires from the top band.

In Fig. 15.15,  $\mathbf{S}$  resends  $m^{\mathbf{S}}$  by using only the first  $\frac{u}{2}$  wires. At a time,  $\mathbf{S}$  tries to send only one block of  $m^{\mathbf{S}}$ . Suppose  $\mathbf{S}$  sends the block  $B_{bc}^{\mathbf{S}}$  to  $\mathbf{R}$  over the wire  $f_{wc}^{\mathbf{S}}$ . If wire  $f_{wc}^{\mathbf{S}}$  is honest then the block will be delivered correctly and  $\mathbf{S}$  will come to know about this through the feedback paths.  $\mathbf{S}$  then asks  $\mathbf{R}$  to increment the block count by authenticating a special signal, where the keys for authentication are selected from  $\mathcal{K}$  and hence will be unknown to the adversary. So except with error probability  $2^{-\Omega(\kappa)}$ ,  $\mathbf{R}$  will correctly receive the signal and will increment the block count.

On the other hand, if  $f_{wc}^{\mathbf{S}}$  is corrupted and delivers incorrect block, then except with error probability  $2^{-\Omega(\kappa)}$ ,  $\mathbf{S}$  will come to know about this. This is because,  $\mathbf{R}$  sends the hash value of the received block where the hash key is selected from  $\mathcal{K}$  and hence will be unknown to the adversary (but will be known to  $\mathbf{S}$ ). In this case,  $\mathbf{S}$  asks  $\mathbf{R}$  to increment the wire count by authenticating a special signal, where the keys for authentication are selected from  $\mathcal{K}$  and hence will be unknown to the adversary. So except with error probability  $2^{-\Omega(\kappa)}$ ,  $\mathbf{R}$  will correctly receive the signal and will increment the wire count.

It is now easy to see that it will take  $\mathcal{O}(u)$  phases, at the end of which either all the blocks of  $m^{\mathbf{S}}$  would have been delivered or both  $\mathbf{S}$  and  $\mathbf{R}$  will come to know that first  $\frac{u}{2} - 1$  wires in the top band are corrupted. In the later case,  $\mathbf{S}$  will re-send the remaining blocks of  $m^{\mathbf{S}}$  in a single phase through wire  $f_{\frac{u}{2}}$ , which is bound to be honest. It is easy to see that the overall communication complexity for resending  $m^{\mathbf{S}}$  is  $\mathcal{O}(|m^{\mathbf{S}}|) = \mathcal{O}(n^3)$  field elements.  $\square$

### 15.7.5 Summary of the Steps of the Protocol

Since there are so many execution sequences possible in protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed, we summarize the steps of protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed in Fig. 15.16.

We now finally state the following theorem:

**Theorem 15.67** *Suppose there exists  $0 \leq u \leq t_b$  wires in the bottom band and  $n = \max(2t_b - u + 1, t_b + 1)$  wires in the top band. Then there exists an  $\mathcal{O}(u)$  SRMT protocol which reliably sends a message containing  $\ell = (t_b - \frac{u}{2} + 1)n^2 = \Theta(n^3)$  field elements by communicating  $\mathcal{O}(\ell) = \mathcal{O}(n^3)$  field elements, tolerating  $\mathcal{A}_b^{static}$ . In terms of bits, the protocol sends  $\Theta(n^3\kappa)$  bits by communicating  $\mathcal{O}(n^3\kappa)$  bits. Thus the protocol achieves reliability with constant factor overhead for sufficiently large message.*

PROOF: Follows from the protocol steps summarized in Fig. 15.16.  $\square$

Once we have an SRMT protocol, which achieves reliability with constant factor overhead, we can easily design a communication optimal SSMT protocol, which we do in the next section.

## 15.8 Communication Optimal SSMT Protocol in Directed Network

We now design an  $\mathcal{O}(u)$  phase SSMT protocol called  $u$ -Optimal-SSMT-Static-Byzantine-Directed, which sends a message  $m^{\mathbf{S}}$  containing  $\ell$  field elements by communicating  $\mathcal{O}(n^3)$  field elements. If the full bottom band is corrupted then  $\ell = \Theta(n^2u)$ , otherwise

$\ell = \Theta(n^2)$ . The protocol uses protocols 6-Pad and  $u$ -Optimal-SRMT-Static-Byzantine-Directed as black box. The protocol is given in Fig. 15.17.

We now state the properties of protocol  $u$ -Optimal-SSMT-Static-Byzantine-Directed.

**Theorem 15.68** *Protocol  $u$ -Optimal-SSMT-Static-Byzantine-Directed is an  $\mathcal{O}(u)$  phase SSMT protocol with a communication complexity of  $\mathcal{O}(n^3)$  field elements, tolerating  $\mathcal{A}_{t_b}^{static}$ . If the entire bottom band is corrupted then the protocol securely sends  $\Theta(n^2u)$  field elements. Otherwise, the protocol securely sends  $\Theta(n^2)$  field elements. In terms of bits, the protocol sends either  $\Theta(n^2u\kappa)$  or  $\Theta(n^3\kappa)$  bits by communicating  $\mathcal{O}(n^3\kappa)$  bits, depending upon whether the entire bottom band is corrupted or not.*

PROOF: The proof is straightforward and follows from the protocol steps and the properties of protocol 6-Pad and  $u$ -Optimal-SRMT-Static-Byzantine-Directed.  $\square$

We next show that the communication complexity of protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed and  $u$ -Optimal-SSMT-Static-Byzantine-Directed is asymptotically optimal. For this, we derive the lower bound on the communication complexity of SRMT and SSMT protocol in directed synchronous network, tolerating tolerating  $\mathcal{A}_{t_b}^{static}$ , in the next section.

## 15.9 Lower Bound on Communication Complexity of SRMT and SSMT

We now derive the lower bound on the communication complexity of SRMT and SSMT protocols. We will then show that our protocols  $u$ -Optimal-SRMT-Static-Byzantine-Directed and  $u$ -Optimal-SSMT-Static-Byzantine-Directed satisfy these bounds asymptotically. We first begin with the lower bound for SRMT protocols.

### 15.9.1 Lower Bound for SRMT Protocols

The lower bound on the communication complexity of SRMT protocols is given by the following theorem:

**Theorem 15.69** *Any SRMT protocol in directed synchronous network, tolerating  $\mathcal{A}_{t_b}^{static}$ , has to communicate  $\Omega(\ell)$  field elements to reliably send a message containing  $\ell$  field elements.*

PROOF: The proof simply follows from the fact that any SRMT protocol has to at least communicate the message.  $\square$

In the light of the above theorem, we state the following theorem:

**Theorem 15.70** *Protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed is a communication optimal SRMT protocol.*

PROOF: Follows from Theorem 15.69 and the fact that  $u$ -Optimal-SRMT-Static-Byzantine-Directed reliably sends a message containing  $\Theta(n^3)$  field elements by communicating  $\mathcal{O}(n^3)$  field elements.  $\square$

In the next section, we derive the lower bound for SSMT protocols.

### 15.9.2 Lower Bound for SSMT Protocols

The derivation of the lower bound on the communication complexity of SSMT protocol is divided into two parts: (a) if the entire bottom band is corrupted and (b) if the entire bottom band is not corrupted. We first consider the case when the entire bottom band is corrupted.

**Theorem 15.71** *Suppose there exists  $u$  wires in the bottom band and  $n$  wires in the top band, where  $u \leq t_b$  and  $n = 2t_b - u + 1$ . Moreover, suppose that the entire bottom band is corrupted. Then any multi phase<sup>6</sup> SSMT protocol to securely send a message  $m$  containing  $\ell$  field elements from  $\mathbb{F}$ , needs to communicate  $\Omega(\frac{n\ell}{u})$  field elements. In terms of bits, the protocol needs to communicate  $\Omega(\frac{n\ell}{u}\kappa)$  bits to securely send  $\ell\kappa$  bits.*

PROOF: Suppose both  $\mathbf{S}$  and  $\mathbf{R}$  in advance knows that the entire bottom band is corrupted. Under this condition, any multiphase SSMT protocol  $\Pi$  to securely send  $m$ , is virtually reduced to a single phase SSMT protocol, where  $\mathbf{S}$  is connected to  $\mathbf{R}$  by  $n = 2t_b - u + 1$  wires, of which at most  $t_b - u$  are corrupted. Since perfect secrecy is required in SSMT, the data sent along the top band in  $\Pi$  must be such that data along any set of  $(t_b - u)$  wires has no information about the secret message  $m$ . Otherwise the adversary will also know the secret message by passively listening the contents of these wires. Similarly, the data sent over any set of  $(n - (t_b - u))$  wires over the top band should have full information about the secret message  $m$ . The latter requirement ensures that even if the adversary simply blocks all the data that he can, the secret message is not lost and therefore the receiver's ability to recover the message is not completely ruled out. We now define the following notations:

1.  $\mathcal{M}$  denotes the message space from where the message  $m$  is selected. In our context,  $\mathcal{M} = \mathbb{F}^\ell$ .
2. For  $i = 1, \dots, n$ ,  $\mathbf{X}_i^m$  denotes the set of all possible transmission in protocol  $\Pi$ , that could occur over wire  $f_i$ , corresponding to message  $m \in \mathcal{M}$ .
3. For  $j \geq i$ ,  $\mathbf{M}_{i,j}^m \subseteq \mathbf{X}_i^m \times \mathbf{X}_{i+1}^m \times \dots \times \mathbf{X}_j^m$  denotes the set of all possible transmission that could occur over wire  $f_i, f_{i+1}, \dots, f_j$  during protocol  $\Pi$ , corresponding to message  $m \in \mathcal{M}$ .
4.  $\mathbf{M}_{i,j} = \bigcup_{m \in \mathcal{M}} \mathbf{M}_{i,j}^m$  and  $\mathbf{X}_i = \bigcup_{m \in \mathcal{M}} \mathbf{X}_i^m$ . We call  $\mathbf{X}_i$  as the *capacity* of wire  $f_i$  and  $\mathbf{M}_{i,j}$  as the *capacity* of the set of wires  $\{f_i, \dots, f_j\}$ .

Now in protocol  $\Pi$ , one element from the set  $\mathbf{X}_i$  is transmitted over wire  $f_i$ , for  $i = 1, \dots, n$ . Moreover, each element of the set  $\mathbf{X}_i$  can be represented by  $\log |\mathbf{X}_i|$  bits. Thus, if we can find out each  $\mathbf{X}_i$ , then the lower bound on the communication complexity of  $\Pi$  will be  $\sum_{i=1}^n \log |\mathbf{X}_i|$  bits. In the sequel, we try to estimate  $\mathbf{X}_i$ .

From the properties of data sent over the top band in protocol  $\Pi$ , the data sent over any set of  $t_b - u$  wires is *independent* of the message. Thus, for any two messages  $m_1, m_2 \in \mathcal{M}$ , it must hold that

$$\mathbf{M}_{t_b-u+1, 2(t_b-u)}^{m_1} = \mathbf{M}_{t_b-u+1, 2(t_b-u)}^{m_2}.$$

*Notice that the above relation must hold for any selection of  $t_b - u$  wires in the top band. We focussed on the set of wires  $\{f_{t_b-u+1}, \dots, f_{2(t_b-u)}\}$  just for simplicity.*

---

<sup>6</sup>Any single phase SSMT protocol in a directed graph is no different from single phase SSMT protocol in undirected graph. The lower bound on the communication complexity of single phase SSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  is given by Theorem 14.3, by substituting  $t_f = t_p = 0$ .

Also, from the properties of data transmitted over the top band in  $\Pi$ , the data transmitted over any set of  $n - (t_b - u)$  wires should have *full* information about the message  $m$  and hence uniquely determine  $m$ . Thus it must also hold that

$$\mathbf{M}_{t_b-u+1,n}^{m_1} \cap \mathbf{M}_{t_b-u+1,n}^{m_2} = \emptyset.$$

We again stress that the above relation must hold for any set of  $n - (t_b - u)$  wires in the top band. We focussed on the set of wires  $\{f_{t_b-u+1}, \dots, f_n\}$  just for simplicity.

As mentioned earlier,  $\mathbf{M}_{t_b-u+1,2(t_b-u)}^m$  will be same for all messages  $m$ . Thus, in order that  $\mathbf{M}_{t_b-u+1,n}^{m_1} \cap \mathbf{M}_{t_b-u+1,n}^{m_2} = \emptyset$  holds, it must be the case that  $\mathbf{M}_{2(t_b-u)+1,n}^m$  is *unique* for every message  $m$ . This implies that

$$|\mathbf{M}_{2(t_b-u)+1,n}| = |\mathcal{M}|.$$

From the definition of  $\mathbf{X}_i$  and  $\mathbf{M}_{i,j}$ , we get

$$\prod_{i=2(t_b-u)+1}^n |\mathbf{X}_i| \geq |\mathbf{M}_{2(t_b-u)+1,n}| \geq |\mathcal{M}|.$$

Let  $g = n - 2(t_b - u)$ . The above inequality holds for any set of  $g$  wires  $\mathcal{D}$  in the top band, where  $|\mathcal{D}| = g$ ; i.e.,  $\prod_{f_i \in \mathcal{D}} |\mathbf{X}_i| \geq |\mathcal{M}|$ . In particular, it holds for every selection  $\mathcal{D}_k$  of set of wires  $\{f_{(kg+1) \bmod n}, f_{(kg+2) \bmod n}, \dots, f_{(kg+g) \bmod n}\}$ , with  $k \in \{0, \dots, n-1\}$ .

If we consider all above  $\mathcal{D}_k$  sets, then each wire in the top band is counted for exactly  $g$  times. Thus, the product of the capacities of all  $\mathcal{D}_k$  yields the capacity of the full top band to the  $g$ -th power, and since each  $\mathcal{D}_k$  has capacity at least  $|\mathcal{M}|$ , we get

$$|\mathcal{M}|^n \leq \prod_{k=0}^{n-1} \prod_{f_j \in \mathcal{D}_k} |\mathbf{X}_j| = (\prod_{i=1}^n |\mathbf{X}_i|)^g,$$

and therefore

$$n \log(|\mathcal{M}|) \leq g \sum_{i=1}^n \log(|\mathbf{X}_i|).$$

As  $\log(|\mathcal{M}|) = \ell \log(|\mathbb{F}|)$ , from the above inequality, we get

$$\sum_{i=1}^n \log(|\mathbf{X}_i|) \geq \left( \frac{n\ell \log(|\mathbb{F}|)}{g} \right) \geq \left( \frac{n\ell \log(|\mathbb{F}|)}{n - 2(t_b - u)} \right).$$

As mentioned earlier,  $\sum_{i=1}^n \log(|\mathbf{X}_i|)$  denotes the lower bound on the communication complexity of protocol  $\Pi$ . From the above inequality, we find that the lower bound on the communication complexity of protocol  $\Pi$  is  $\Omega\left(\frac{n\ell \log(|\mathbb{F}|)}{n - 2(t_b - u)}\right) = \Omega\left(\frac{n\ell}{n - 2(t_b - u)} \kappa\right)$  bits. Now each field element from  $\mathbb{F}$  can be represented by  $\kappa$  bits. Thus the lower bound on the communication complexity of protocol  $\Pi$  is  $\Omega\left(\frac{n\ell}{n - 2(t_b - u)}\right) = \Omega\left(\frac{n\ell}{u}\right)$  field elements, as  $n = 2t_b - u + 1$ .  $\square$

We now proceed to the second case when the entire bottom band is not corrupted. The lower bound on the communication complexity of SSMT protocol for this case is given by the following theorem:

**Theorem 15.72** *Suppose there exists  $u \leq t_b$  wires in the bottom band and  $n = 2t_b - u + 1$  wires in the top band. Moreover, suppose that the entire bottom band is not corrupted. Then any multiphase SSMT protocol to securely send a message  $m$  containing  $\ell$  field elements from  $\mathbb{F}$ , needs to communicate  $\Omega(n\ell)$  field elements. In terms of bits, the protocol needs to communicate  $\Omega(n\ell\kappa)$  bits to securely send  $\ell\kappa$  bits.*

PROOF: Suppose there exists  $u \leq t_b$  wires in the bottom band and  $n = 2t_b - u + 1$  wires in the top band. Let  $N = (n + u) = 2t_b + 1$ . Now  $n = \Theta(t_b)$ , as there exists at least  $t_b + 1$  wires in the top band. Also  $N = \Theta(t_b)$ . Let  $\Pi$  be a multiphase SSMT protocol over the  $N$  wires to securely transmit a message  $m$  containing  $\ell$  field elements from  $\mathbb{F}$ . Then in any execution of  $\Pi$ , the data exchanged along any set of  $t_b$  wires (including top and bottom band) must be independent of  $m$ . Otherwise, adversary can passively listen these wires and will know  $m$ , which violates perfect secrecy condition of  $\Pi$ . On the other hand, data exchanged along any set of  $N - t_b$  wires (including top and bottom band) should have complete information about the message. The latter requirement ensures that even if the adversary simply blocks all the data that he can, the secret message is not lost and therefore the receiver's ability to recover the message is not completely ruled out. Let the  $N$  wires between  $\mathbf{S}$  and  $\mathbf{R}$  be denoted by  $w_1, \dots, w_N$ . Out of these  $N$  wires, the first  $n$  wires are the wires from the top band, which are directed from  $\mathbf{S}$  to  $\mathbf{R}$ . On the other hand, remaining  $N - n = u$  wires are from the bottom band, which are directed from  $\mathbf{R}$  to  $\mathbf{S}$ .

We now define the following notations:

1.  $\mathcal{M}$  denotes the message space from where the message  $m$  is selected. In our context,  $\mathcal{M} = \mathbb{F}^\ell$ .
2. For  $i = 1, \dots, N$ ,  $\mathbf{X}_i^m$  denotes the set of all possible transmission in protocol  $\Pi$ , that could occur over wire  $w_i$ , corresponding to message  $m \in \mathcal{M}$ .
3. For  $j \geq i$ ,  $\mathbf{M}_{i,j}^m \subseteq \mathbf{X}_i^m \times \mathbf{X}_{i+1}^m \times \dots \times \mathbf{X}_j^m$  denotes the set of all possible transmission that could occur over wire  $w_i, w_{i+1}, \dots, w_j$  during protocol  $\Pi$ , corresponding to message  $m \in \mathcal{M}$ .
4.  $\mathbf{M}_{i,j} = \bigcup_{m \in \mathcal{M}} \mathbf{M}_{i,j}^m$  and  $\mathbf{X}_i = \bigcup_{m \in \mathcal{M}} \mathbf{X}_i^m$ . We call  $\mathbf{X}_i$  as the *capacity* of wire  $w_i$  and  $\mathbf{M}_{i,j}$  as the *capacity* of the set of wires  $\{w_i, \dots, w_j\}$ .

Now in protocol  $\Pi$ , one element from the set  $\mathbf{X}_i$  is transmitted over wire  $w_i$ , for  $i = 1, \dots, N$ . Moreover, each element of the set  $\mathbf{X}_i$  can be represented by  $\log |\mathbf{X}_i|$  bits. Thus, if we can find out each  $\mathbf{X}_i$ , then the lower bound on the communication complexity of  $\Pi$  will be  $\sum_{i=1}^N \log |\mathbf{X}_i|$  bits. In the sequel, we try to estimate  $\mathbf{X}_i$ .

From the properties of protocol  $\Pi$ , the data sent over any set of  $t_b$  wires is *independent* of the message. Thus, for any two messages  $m_1, m_2 \in \mathcal{M}$ , it must hold that

$$\mathbf{M}_{t_b+1, 2t_b}^{m_1} = \mathbf{M}_{t_b+1, 2t_b}^{m_2}.$$

*Notice that the relation above must hold for any selection of  $t_b$  wires out of the  $N$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ . We focussed on the set of wires  $\{w_{t_b+1}, \dots, w_{2t_b}\}$  just for simplicity.*

Also, from the properties of protocol  $\Pi$ , the data transmitted over any set of  $N - t_b$  wires should have *full* information about the message  $m$  and hence uniquely determine  $m$ . Thus it must also hold that

$$\mathbf{M}_{t_b+1, N}^{m_1} \cap \mathbf{M}_{t_b+1, N}^{m_2} = \emptyset.$$

*We again stress that the above relation must hold for any set of  $N - t_b$  wires out of the  $N$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ . We focussed on the set of wires  $\{w_{t_b+1}, \dots, w_N\}$  just for simplicity.*

As mentioned earlier,  $\mathbf{M}_{t_b+1, 2t_b}^m$  will be same for all messages  $m$ . Thus, in order that  $\mathbf{M}_{t_b+1, N}^{m_1} \cap \mathbf{M}_{t_b+1, N}^{m_2} = \emptyset$  holds, it must be the case that  $\mathbf{M}_{t_b+1, N}^m$  is *unique* for every message  $m$ . This implies that

$$|\mathbf{M}_{2t_b+1, N}| = |\mathcal{M}|.$$

From the definition of  $\mathbf{X}_i$  and  $\mathbf{M}_{i,j}$ , we get

$$\prod_{i=2t_b+1}^N |\mathbf{X}_i| \geq |\mathbf{M}_{2t_b+1,N}| \geq |\mathcal{M}|.$$

Let  $g = N - 2t_b$ . The above inequality holds for any set of  $g$  wires  $\mathcal{D}$ , where  $|\mathcal{D}| = g$ ; i.e.,  $\prod_{w_i \in \mathcal{D}} |\mathbf{X}_i| \geq |\mathcal{M}|$ . In particular, it holds for every selection  $\mathcal{D}_k$  of set of wires  $\{w_{(kg+1) \bmod N}, w_{(kg+2) \bmod N}, \dots, w_{(kg+g) \bmod N}\}$ , with  $k \in \{0, \dots, N-1\}$ .

If we consider all above  $\mathcal{D}_k$  sets, then each wire is counted exactly  $g$  times. Thus, the product of the capacities of all  $\mathcal{D}_k$  yields the capacity of the full top band and bottom band to the  $g$ -th power, and since each  $\mathcal{D}_k$  has capacity at least  $|\mathcal{M}|$ , we get

$$|\mathcal{M}|^N \leq \prod_{k=0}^{N-1} \prod_{w_j \in \mathcal{D}_k} |\mathbf{X}_j| = \left( \prod_{i=1}^N |\mathbf{X}_i| \right)^g,$$

and therefore

$$N \log(|\mathcal{M}|) \leq g \sum_{i=1}^N \log(|\mathbf{X}_i|).$$

As  $\log(|\mathcal{M}|) = \ell \log(|\mathbb{F}|)$ , from the above inequality, we get

$$\sum_{i=1}^N \log(|\mathbf{X}_i|) \geq \left( \frac{N \ell \log(|\mathbb{F}|)}{g} \right) \geq \left( \frac{N \ell \log(|\mathbb{F}|)}{N - 2t_b} \right).$$

As mentioned earlier,  $\sum_{i=1}^N \log(|\mathbf{X}_i|)$  denotes the lower bound on the communication complexity of protocol  $\Pi$ . From the above inequality, we find that the lower bound on the communication complexity of protocol  $\Pi$  is  $\Omega\left(\frac{N \ell \log(|\mathbb{F}|)}{N - 2t_b}\right) = \Omega(N \ell \kappa)$  bits, as  $N = 2t_b + 1$ . Now each field element from  $\mathbb{F}$  can be represented by  $\kappa$  bits. Thus the lower bound on the communication complexity of protocol  $\Pi$  is  $\Omega(N \ell) = \Omega(n \ell)$  field elements, as  $N = \Theta(t_b) = \Theta(n)$ .  $\square$

Now in the light of previous two theorems, we can state the following theorem:

**Theorem 15.73** *Protocol  $u$ -Optimal-SSMT-Static-Byzantine-Directed is a communication optimal SSMT protocol.*

PROOF: From Theorem 15.68, if the entire bottom band is corrupted then protocol  $u$ -Optimal-SSMT-Static-Byzantine-Directed securely sends  $\ell = \Theta(n^2 u)$  field elements by communicating  $\mathcal{O}(n^3)$  field elements. From Theorem 15.71, if the entire bottom band is corrupted, then any multiphase SSMT protocol has to communicate  $\Omega\left(\frac{n \ell}{u}\right) = \Omega(n^3)$  field elements to securely send  $\ell = \Theta(n^2 u)$  field elements. So if the entire bottom band is corrupted then protocol  $u$ -Optimal-SSMT-Static-Byzantine-Directed is a communication optimal SSMT protocol.

If the entire bottom band is not corrupted then protocol  $u$ -Optimal-SSMT-Static-Byzantine-Directed securely sends  $\ell = \Theta(n^2)$  field elements by communicating  $\mathcal{O}(n^3)$  field elements. From Theorem 15.72, if the entire bottom band is not corrupted, then any multiphase SSMT protocol has to communicate  $\Omega(n \ell) = \Omega(n^3)$  field elements to securely send  $\ell = \Theta(n^2)$  field elements. So if the entire bottom band is not corrupted then also protocol  $u$ -Optimal-SSMT-Static-Byzantine-Directed is a communication optimal SSMT protocol.  $\square$

## 15.10 Concluding Remarks and Open Problems

In this chapter, we have shown that the existing SRMT and SSMT protocols of [24, 87, 54] in directed networks, tolerating  $\mathcal{A}_{t_b}^{static}$  are inefficient. We then proposed new

SRMT and SSMT protocols in directed networks, tolerating  $\mathcal{A}_{t_b}^{static}$ , which are also communication optimal. This completely resolves the issue of OPTIMALITY of SRMT and SSMT in directed synchronous network tolerating  $\mathcal{A}_{t_b}^{static}$ . This chapter leaves few open problems, which are as follows:

**Open Problem 17** *Our communication optimal SRMT and SSMT protocol require  $\mathcal{O}(u)$  phases. It would be interesting to come up with communication optimal SRMT and SSMT protocols with less phase complexity.*

**Open Problem 18** *Our SRMT and SSMT protocol is communication optimal only for messages of some specific length. It would be interesting to design SRMT and SSMT protocols in directed networks tolerating  $\mathcal{A}_{t_b}^{static}$ , which are communication optimal for messages of any length.*

Till now, all our discussion in this thesis focussed on synchronous network. We now proceed to the last part of our thesis, where we discuss about PSMT and SSMT in asynchronous networks.

Figure 15.13: Execution of  $u$ -Optimal-SRMT-Static-Byzantine-Directed If **S** Receives  $\frac{u}{2} + 1$  Identical Conflict Lists and  $|L_{fault}^S| \leq (t_b - \frac{u}{2})$

**Phase III: S to R:**

1. By selecting two keys from the global key set  $\mathcal{K}$ , **S** authenticates special "terminate2" signal using  $URauth$  function and sends it to **R** through the top band.
2. By selecting  $2|L_{fault}^S|$  keys and  $10|\mathcal{Y}^S|$  keys from global key set  $\mathcal{K}$ , **S** authenticates each element of  $L_{fault}^S$  and  $\mathcal{Y}^S$  respectively using  $URauth$  function. Let  $L_{fault_{auth}}^S$  and  $\mathcal{Y}_{auth}^S$  denote the set of corresponding authenticated values.
3. **S** then sends  $(\mathcal{Y}^S, L_{fault}^S, \mathcal{Y}_{auth}^S, L_{fault_{auth}}^S)$  to **R** through the top band and terminates protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed.

**Computation by R at the End of Phase III:**

1. With very high probability, **R** correctly receives "terminate2" signal.
2. Let **R** receive  $(\mathcal{Y}_i^R, L_{fault_i}^R, \mathcal{Y}_{i,auth}^R, L_{fault_{i,auth}}^R)$  from **S** along wire  $f_i$ , for  $i = 1, \dots, n$ . From these values, **R** now tries to find out whether **S** has correctly received the original  $\mathcal{Y}^R$  over more than  $\frac{u}{2} + 1$  wires during **Phase I**, and if yes, then the corresponding  $L_{fault}^S$ . For this, **R** does the following computation:
  - (a) For each  $i = 1, \dots, n$ , **R** checks  $\mathcal{Y}_i^R \stackrel{?}{=} \mathcal{Y}^R$  and  $|L_{fault_i}^R| \leq (t_b - \frac{u}{2})$ . If any of these test fails, then **R** neglects all the values received along  $f_i$ . Otherwise, **R** applies the  $URauth$  function to each element of  $\mathcal{Y}_i^R$  by using the same keys from  $\mathcal{K}$ , which were used by **S** to authenticate  $\mathcal{Y}^S$  and computes the set  $\mathcal{Y}'_{i,auth}{}^R$ . Similarly, **R** applies  $URauth$  function to each element of  $L_{fault_i}^R$  by using the same keys from  $\mathcal{K}$ , which were used by **S** to authenticate  $L_{fault}^S$  and computes the set  $L'_{fault_{i,auth}}{}^R$ . **R** then checks  $\mathcal{Y}'_{i,auth}{}^R \stackrel{?}{=} \mathcal{Y}_{i,auth}^R$  and  $L'_{fault_{i,auth}}{}^R \stackrel{?}{=} L_{fault_{i,auth}}^R$ . If the test fails then **R** discards the values received along  $f_i$ .
  - (b) If all the wires in the top band are discarded by **R** during previous step, then **R** concludes that **S** has not received original  $\mathcal{Y}^R$  over more than  $\frac{u}{2} + 1$  wires during **Phase II**, which further implies that at most  $t_b - \frac{u}{2} - 1$  wires were corrupted in the top band during **Phase I**. So **R** recovers  $m^R$  by using the  $F_i^R$ 's received over the wires in  $\mathcal{P}^R$  during **Phase I** and terminates  $u$ -Optimal-SRMT-Static-Byzantine-Directed.
  - (c) If there exists an  $i \in \{1, 2, \dots, n\}$  such that  $\mathcal{Y}_i^R = \mathcal{Y}^R$ ,  $|L_{fault_i}^R| \leq (t_b - \frac{u}{2})$ ,  $\mathcal{Y}'_{i,auth}{}^R = \mathcal{Y}_{i,auth}^R$  and  $L'_{fault_{i,auth}}{}^R = L_{fault_{i,auth}}^R$ , then **R** concludes that **S** has correctly received original  $\mathcal{Y}^R$  over more than  $\frac{u}{2} + 1$  wires during **Phase II** and  $L_{fault_i}^R$  is the corresponding  $L_{fault}$  sent by **S**. Now by using the  $F_i^R$ 's received over the wires not in  $L_{fault_i}^R$ , **R** recovers  $m^R$  and terminates  $u$ -Optimal-SRMT-Static-Byzantine-Directed.

Figure 15.14: Execution of  $u$ -Optimal-SRMT-Static-Byzantine-Directed If  $\mathbf{S}$  Receives  $\frac{u}{2} + 1$  Identical Conflict Lists and  $|L_{fault}^{\mathbf{S}}| \geq (t_b - \frac{u}{2}) + 1$

**Phase III: S to R:** Same as in Fig. 15.13, except that  $\mathbf{S}$  does not send "terminate" signal. Moreover, here  $|L_{fault}^{\mathbf{S}}| \geq (t_b - \frac{u}{2}) + 1$ .

**Phase IV: R to S:**

1. Let  $\mathbf{R}$  receive  $(\mathcal{Y}_i^{\mathbf{R}}, L_{fault_i}^{\mathbf{R}}, \mathcal{Y}_{i,auth}^{\mathbf{R}}, L_{fault_{i,auth}}^{\mathbf{R}})$  from  $\mathbf{S}$  along wire  $f_i$ , for  $i = 1, \dots, n$ . From these values,  $\mathbf{R}$  tries to find out whether  $\mathbf{S}$  has correctly received the original  $\mathcal{Y}^{\mathbf{R}}$  over more than  $\frac{u}{2} + 1$  wires during **Phase I**, and if yes, then the corresponding  $L_{fault}^{\mathbf{S}}$ . For this,  $\mathbf{R}$  does the same computation as done by  $\mathbf{R}$  at the end of **Phase III** in Fig. 15.13. However, now instead of checking  $|L_{fault_i}^{\mathbf{R}}| \leq (t_b - \frac{u}{2})$ ,  $\mathbf{R}$  checks  $|L_{fault_i}^{\mathbf{R}}| \geq (t_b - \frac{u}{2} + 1)$ .
2. If after the checking in the previous steps, all the wires in the top band are discarded by  $\mathbf{R}$  then  $\mathbf{R}$  concludes that  $\mathbf{S}$  has not received original  $\mathcal{Y}^{\mathbf{R}}$  over more than  $\frac{u}{2} + 1$  wires during **Phase II**, which further implies that at most  $t_b - \frac{u}{2} - 1$  wires were corrupted in the top band during **Phase I**. So  $\mathbf{R}$  recovers  $m^{\mathbf{R}}$  by using the  $F_i^{\mathbf{R}}$ 's received over the wires in  $\mathcal{P}^{\mathbf{R}}$  during **Phase I**. Moreover,  $\mathbf{R}$  computes  $response_1 = URauth("terminate"; k_1, k_2)$ , where  $k_1, k_2$  are selected from  $\mathcal{K}$ . Finally  $\mathbf{R}$  asks  $\mathbf{S}$  to terminate  $u$ -Optimal-SRMT-Static-Byzantine-Directed by sending  $(\text{"terminate"}, response_1)$  over the bottom band and terminates  $u$ -Optimal-SRMT-Static-Byzantine-Directed.
3. If  $\mathbf{R}$  finds an  $i \in \{1, 2, \dots, n\}$  such that  $\mathcal{Y}_i^{\mathbf{R}} = \mathcal{Y}^{\mathbf{R}}$ ,  $|L_{fault_i}^{\mathbf{R}}| \geq (t_b - \frac{u}{2} + 1)$ ,  $\mathcal{Y}_{i,auth}^{\mathbf{R}} = \mathcal{Y}_{i,auth}^{\mathbf{R}}$  and  $L_{fault_{i,auth}}^{\mathbf{R}} = L_{fault_{i,auth}}^{\mathbf{R}}$ , then  $\mathbf{R}$  concludes that  $\mathbf{S}$  has correctly received original  $\mathcal{Y}^{\mathbf{R}}$  over more than  $\frac{u}{2} + 1$  wires during **Phase II** and  $L_{fault_i}^{\mathbf{R}}$  is the corresponding  $L_{fault}$  sent by  $\mathbf{S}$ . So  $\mathbf{R}$  removes the wires in  $L_{fault_i}^{\mathbf{R}}$  from his view for further computation and communication. Now by selecting  $k_1, k_2$  from  $\mathcal{K}$  as authentication keys,  $\mathbf{R}$  computes  $response_2 = URauth("continue"; k_1, k_2)$  where "continue" is a unique pre-defined special signal.  $\mathbf{R}$  then send the tuple  $(\text{"continue"}, response_2)$  to  $\mathbf{S}$  through the bottom band.

**Computation by S at the end of Phase IV:**

1.  $\mathbf{S}$  checks whether it is getting any 2-tuple identically over at least  $\frac{u}{2} + 1$  wires. If not, then  $\mathbf{S}$  concludes that  $\mathbf{R}$  has recovered  $m^{\mathbf{R}}$  at the end of **Phase III** and terminates  $u$ -Optimal-SRMT-Static-Byzantine-Directed.
2. If  $\mathbf{S}$  receives a 2-tuple say  $(x_1^{\mathbf{S}}, y_1^{\mathbf{S}})$  over  $\frac{u}{2} + 1$  wires, then  $\mathbf{S}$  verifies  $y_1^{\mathbf{S}} \stackrel{?}{=} URauth(x_1^{\mathbf{S}}; k_1, k_2)$ , where  $k_1$  and  $k_2$  are the keys from  $\mathcal{K}$ . If the test fails, then  $\mathbf{S}$  again concludes that  $\mathbf{R}$  has recovered  $m^{\mathbf{R}}$  at the end of **Phase III** and terminates  $u$ -Optimal-SRMT-Static-Byzantine-Directed.
3. If the test in the previous step succeeds then  $\mathbf{S}$  further checks  $x_1^{\mathbf{S}} \stackrel{?}{=} \text{"terminate"}$ . If yes, then  $\mathbf{S}$  again concludes that  $\mathbf{R}$  has recovered  $m^{\mathbf{R}}$  at the end of **Phase III** and terminates  $u$ -Optimal-SRMT-Static-Byzantine-Directed. On the other hand, if  $x_1^{\mathbf{S}} \stackrel{?}{=} \text{"continue"}$  then  $\mathbf{S}$  concludes that  $\mathcal{Y}^{\mathbf{S}}$  was indeed sent by  $\mathbf{R}$  and hence the wires in  $L_{fault}^{\mathbf{S}}$  are indeed corrupted. So  $\mathbf{S}$  removes them from his view point for further computation and communication and further continues the protocol.

Figure 15.15: Execution of  $u$ -Optimal-SRMT-Static-Byzantine-Directed If  $\mathbf{S}$  Receives "continue" Signal Through the Majority Wires in the Bottom Band at the End of **Phase IV**

**Assumption:** Without loss of generality, we assume that the first  $\frac{u}{2}$  wires are still in the consideration of  $\mathbf{S}$  and  $\mathbf{R}$ .

1.  $\mathbf{S}$  divides  $m^{\mathbf{S}}$  into blocks  $B_1^{\mathbf{S}}, \dots, B_{\frac{u}{2}}^{\mathbf{S}}$ , each of size  $\frac{|m^{\mathbf{S}}|}{\frac{u}{2}}$  field elements.
2.  $\mathbf{S}$  and  $\mathbf{R}$  initializes variables  $wc^{\mathbf{S}} = 1, bc^{\mathbf{S}} = 1$  and  $wc^{\mathbf{R}} = 1, bc^{\mathbf{R}} = 1$  respectively. Here  $wc$  stands for wire count and  $bc$  stands for block count.
3.  $\mathbf{S}$  and  $\mathbf{R}$  now executes the following steps:
  - (a) While  $(wc^{\mathbf{S}} \leq \frac{u}{2} - 1)$  and (all the blocks of  $m^{\mathbf{S}}$  are not delivered to  $\mathbf{R}$ ) do the following:
    - i.  $\mathbf{S}$  sends the block  $B_{bc^{\mathbf{S}}}^{\mathbf{S}}$  to  $\mathbf{R}$  *only* over the wire  $f_{wc^{\mathbf{S}}}$  in the top band.
    - ii. Let  $\mathbf{R}$  receive  $B_{bc^{\mathbf{R}}}^{\mathbf{R}}$  along wire  $f_{wc^{\mathbf{R}}}$ . Now by selecting a hash key  $k_{bc}$  from the set  $\mathcal{K}$ ,  $\mathbf{R}$  computes  $x_{bc}^{\mathbf{R}} = \text{hash}(k_{bc}; B_{bc^{\mathbf{R}}}^{\mathbf{R}})$  and sends  $x_{bc}^{\mathbf{R}}$  to  $\mathbf{S}$  through the entire bottom band.
    - iii.  $\mathbf{S}$  correctly receives  $x_{bc}^{\mathbf{R}}$  through at least  $\frac{u}{2} + 1$  wires (recall that in this case majority wires in bottom band are honest) and verifies  $x_{bc}^{\mathbf{R}} \stackrel{?}{=} \text{hash}(k_{bc}; B_{bc^{\mathbf{S}}}^{\mathbf{S}})$ .
    - iv. If the test fails then  $\mathbf{S}$  concludes that wire  $f_{wc^{\mathbf{S}}}$  has delivered incorrect  $B_{bc^{\mathbf{S}}}^{\mathbf{S}}$  to  $\mathbf{R}$ . So  $\mathbf{S}$  does the following:
      - A.  $\mathbf{S}$  increments  $wc^{\mathbf{S}}$  by one.
      - B.  $\mathbf{S}$  authenticates a unique, special, pre-defined signal "increment-wire" by using two keys from the set  $\mathcal{K}$  and sends the authenticated signal to  $\mathbf{R}$  through the top band.
      - C.  $\mathbf{R}$  correctly receives the signal with very high probability and accordingly increments  $wc^{\mathbf{R}}$  by one.
    - v. If the test succeeds then  $\mathbf{S}$  concludes that wire  $f_{wc^{\mathbf{S}}}$  has delivered correct  $B_{bc^{\mathbf{S}}}^{\mathbf{S}}$  to  $\mathbf{R}$ . So  $\mathbf{S}$  does the following:
      - A.  $\mathbf{S}$  increments  $bc^{\mathbf{S}}$  by one.
      - B.  $\mathbf{S}$  authenticates a unique, pre-defined, special "increment-block" signal by using keys from the set  $\mathcal{K}$  and sends it to  $\mathbf{R}$  through the top band.
      - C.  $\mathbf{R}$  correctly receives the signal with very high probability and accordingly increments  $bc^{\mathbf{R}}$  by one.
  - (b) If all the blocks of  $m^{\mathbf{S}}$  are delivered then both  $\mathbf{S}$  and  $\mathbf{R}$  terminates. Otherwise  $\mathbf{S}$  concatenates all the remaining blocks of  $m^{\mathbf{S}}$  and sends it to  $\mathbf{R}$  through wire  $f_{\frac{u}{2}}$  and terminates.  $\mathbf{R}$  correctly receives these blocks and terminates.

Figure 15.16: Summary of the Steps Executed in Protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed

1. **S** and **R** executes first two phases as shown in Fig. 15.10. This requires a communication complexity of  $\mathcal{O}(n^3)$  field elements (see Claim 15.58).
2. If a unique conflict list is not received by **S** at the end of **Phase II** through at least  $\frac{u}{2} + 1$  wires in the bottom band then **S** and **R** executes the steps given in Fig. 15.11. In this case, protocol terminates at the end of **Phase III** and the overall communication complexity of the protocol is  $\mathcal{O}(n^3)$  field elements (see Lemma 15.61).
3. If a unique conflict list  $\mathcal{Y}^{\mathbf{S}}$  is received by **S** at the end of **Phase II** through at least  $\frac{u}{2} + 1$  wires in the bottom band then **S** computes  $L_{fault}^{\mathbf{S}}$  from  $\mathcal{Y}^{\mathbf{S}}$  by following the steps given in Fig. 15.12.
4. If  $|L_{fault}^{\mathbf{S}}| \leq (t_b - \frac{u}{2})$  then **S** and **R** executes the steps given in Fig. 15.13. In this case, protocol terminates at the end of **Phase III** and the overall communication complexity of the protocol is  $\mathcal{O}(n^3)$  field elements (see Lemma 15.64).
5. If  $|L_{fault}^{\mathbf{S}}| \geq (t_b - \frac{u}{2} + 1)$  then **S** and **R** executes the steps given in Fig. 15.14. Now there are two possible cases:
  - (a) If **R** terminates the protocol at the end of **Phase III** then **S** will also terminate the protocol at the end of **Phase IV**. In this case, the communication complexity of the protocol will be  $\mathcal{O}(n^3)$  field elements (see Lemma 15.65).
  - (b) If **R** decides to continue the protocol then at the end of **Phase IV**, **S** will also come to know this. In this case, **S** and **R** executes the steps given in Fig. 15.15. The protocol will require  $\mathcal{O}(u)$  phases and a communication complexity of  $\mathcal{O}(n^3)$  field elements (see Lemma 15.66).

Figure 15.17: An  $\mathcal{O}(u)$  Phase Communication Optimal SSMT Protocol  $u$ -Optimal-SSMT-Static-Byzantine-Directed

1. **S** and **R** securely establishes a random, non-zero one time pad  $Pad$  by executing the six phase protocol 6-Pad. If the entire bottom band is corrupted then the size of the pad is  $\Theta(n^2u)$  field elements, otherwise the size of the pad is  $\Theta(n^2)$  field elements.
2. If  $Pad$  is of size  $\Theta(n^2u)$  field elements, then **S** selects a secret message  $m^{\mathbf{S}}$  containing  $\Theta(n^2u)$  field elements. **S** then computes  $C = m^{\mathbf{S}} \oplus Pad$ . **S** then appends some extra field elements to  $C$  from  $\mathbb{F}$ , such that  $C$  contains  $\Theta(n^3)$  field elements. The appended elements are randomly selected from  $\mathbb{F}$ . Finally, **S** reliably sends  $C$  to **R** by executing the protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed (the random elements are appended to  $C$  so that  $C$  contains  $\Theta(n^3)$  field elements. This is because protocol  $u$ -Optimal-SRMT-Static-Byzantine-Directed requires the minimum message size to be  $\Theta(n^3)$  field elements). **R** correctly receives  $C$  with very high probability. **R** then remove the last elements from  $C$ , such that  $C$  contains  $\Theta(n^2u)$  field elements. Finally **R** computes  $m^{\mathbf{S}} = C \oplus Pad$  and terminates the protocol.
3. If  $Pad$  contains  $\Theta(n^2)$  field elements, then **S** and **R** does the same computation as above, except that  $m^{\mathbf{S}}$  and original  $C$  will be of size  $\Theta(n^2)$  field elements.

## Part III

# Results for PSMT and SSMT in Asynchronous Network

## Chapter 16

# PSMT and SSMT in Asynchronous Network Tolerating Static Byzantine Adversary

The existing results for PSMT and SSMT and all the results that are discussed till now in this thesis assume the underlying network to be *synchronous*. Thus, if  $\mathbf{S}$  ( $\mathbf{R}$ ) sends some information along a wire, then it is assumed that  $\mathbf{R}$  ( $\mathbf{S}$ ) will get the information (possibly corrupted) along the wire after a *fixed* interval of time. Though theoretically impressive, this is a very strong assumption because the delay in the arrival of a single message will affect the overall security of the protocol. A typical large network like the Internet can be modelled more accurately by asynchronous networks than synchronous networks. The inherent difficulty in designing a protocol in asynchronous network comes from the fact that *we cannot distinguish between a slow sender and a corrupted sender*. Thus a receiver cannot wait to receive message along all the wires, as waiting for all of them may turn out to be endless. In the literature, very little attention has been paid to the study of PSMT and SSMT protocols in asynchronous network due to its complexity. This motivates us to study PSMT and SSMT protocols in asynchronous networks.

In this chapter, we study PSMT in SSMT in asynchronous network tolerating threshold static Byzantine adversary  $\mathcal{A}_{t_b}^{static}$ . To the best of our knowledge, the only known PSMT protocol in asynchronous network tolerating  $\mathcal{A}_{t_b}^{static}$  is due to [69]. However, in this chapter, we show that the PSMT protocol of [69] does not provide perfect security. We then give the characterization for PSMT and SSMT in asynchronous network tolerating  $\mathcal{A}_{t_b}^{static}$ , thus completely resolving the issue of POSSIBILITY. The most interesting fact brought forth by our characterization is the following: *our characterization shows that asynchrony of the network demands higher connectivity of the network for the existence of PSMT protocols. On the other hand, asynchrony of the network does not demand higher connectivity of the network for SSMT protocols.*

We now define the asynchronous network model and adversary settings used in this chapter.

## 16.1 Asynchronous Network Model

We consider a completely asynchronous network  $\mathcal{N}$ , where  $\mathbf{S}$ ,  $\mathbf{R}$  are two special nodes in  $\mathcal{N}$ . All the nodes in  $\mathcal{N}$  are modelled as probabilistic interactive Turing Machines, where randomization is achieved through random coins. The corruption in the network is modelled by a *centralized adversary*  $\mathcal{A}_{t_b}^{static}$ , who has *unbounded computing power* and can actively control at most  $t_b$  nodes in the network, excluding  $\mathbf{S}$  and  $\mathbf{R}$  in Byzantine fashion.

Following the approach of Dolev et al. [28], we abstract the network and assume that  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n$  vertex disjoint paths, called *wires*, of which at most  $t_b$  could be actively controlled by  $\mathcal{A}_{t_b}^{static}$  in Byzantine fashion. Moreover, we consider two extreme cases:

1. When all the  $n$  wires are directed from  $\mathbf{S}$  to  $\mathbf{R}$ , thus not allowing any interaction between  $\mathbf{S}$  and  $\mathbf{R}$ ;
2. When all the  $n$  wires are bi-directional, thus allowing interaction between  $\mathbf{S}$  and  $\mathbf{R}$ .

To model the asynchrony in the network, we assume that the adversary can schedule the message delivery along *every* wire; i.e., he can determine the time delay of all the messages along all the  $n$  wires. However, *adversary can only schedule the messages sent along honest wires, without having any access to them*. Moreover, the message sent over an honest wire will be eventually delivered. If a wire is under the control of  $\mathcal{A}_{t_b}^{static}$ , then  $\mathcal{A}_{t_b}^{static}$  may indefinitely block the communication along the wire. So the receiver may have to wait indefinitely for the message(s) along that wire. Hence the *receiver can not distinguish between honest wires which are slow (due to the malicious scheduling of messages by  $\mathcal{A}_{t_b}^{static}$  on these wires) and corrupted wires which withhold/does not send information at all*.

In our protocols,  $\mathbf{S}$  and  $\mathbf{R}$  does computation over a field  $\mathbb{F}$ , where  $\mathbb{F}$  is a finite field of prime order. For PSMT protocols, the only restriction on  $\mathbb{F}$  is that  $|\mathbb{F}| > n$ . On the other hand, for SSMT protocol,  $\mathbb{F} = GF(2^\kappa)$ , where  $\kappa$  is the error parameter of the protocol. If some  $x \in \mathbb{F}$  is sent through all the wires, then it is said to be broadcasted. If  $x$  is broadcasted over at least  $2t_b + 1$  wires, then receiver will always correctly recover it. This is because out of the  $2t_b + 1$  wires, at least  $t_b + 1$  will be honest and will eventually deliver  $x$ . So the receiver can wait for a value which is received identically over at least  $t_b + 1$  wires. We now define *asynchronous perfectly secure message transmission* (APSMT) and *asynchronous statistically secure message transmission* (ASSMT).

Let the message to be transmitted securely be drawn from  $\mathbb{F}$  and  $\Gamma$  denote the underlying probability distribution on  $\mathbb{F}$ . We define the view of a node  $P_j \in \mathcal{N}$ , at any point of the execution of a protocol  $\Pi$  for secure message transmission, to be the information that  $P_j$  can get from its local input to the protocol (if any), all the messages that  $P_j$  had earlier sent or received, the protocol code executed by  $P_j$  and random coins of  $P_j$ . The view of  $\mathcal{A}_{t_b}^{static}$  at any point of the execution of  $\Pi$  is defined as all the information that  $\mathcal{A}_{t_b}^{static}$  can get from the views of all the nodes corrupted by  $\mathcal{A}_{t_b}^{static}$  (i.e. all the information that these nodes can commonly compute from their views). For message  $m \in \mathbb{F}$ , any adversary characterized by  $\mathcal{A}_{t_b}^{static}$  and any protocol  $\Pi$  for secure message transmission, let  $\widehat{\Gamma}(\mathcal{A}_{t_b}^{static}, m, \Pi)$  denote the probability distribution on the view of the adversary  $\mathcal{A}_{t_b}^{static}$  at the end of the execution of  $\Pi$ .

**Definition 16.1 (APSMT)** *A protocol  $\Pi$  is said to facilitate asynchronous perfectly*

secure message transmission (APSMT) if for any message  $m$  drawn from  $\mathbb{F}$  and for every adversary  $\mathcal{A}_{t_b}^{static}$ , the following conditions are satisfied:

1. *Perfect Secrecy:*  $\widehat{\Gamma}(\mathcal{A}_{t_b}^{static}, m, \Pi) = \widehat{\Gamma}(\mathcal{A}_{t_b}^{static}, m', \Pi)$ , for all  $m' \neq m$ . That is, the two distributions are identical irrespective of the message transmitted.
2. *Perfect Reliability:*  $\mathbf{R}$  should receive  $m$  correctly, without any error.
3. *Termination:*  $\mathbf{R}$  should eventually terminate the protocol.

**Definition 16.2 (ASSMT)** A protocol  $\Pi$  is said to facilitate asynchronous statistically secure message transmission (ASSMT) if a negligible error probability of  $2^{-\Omega(\kappa)}$  can be tolerated with respect to the Perfect Reliability condition of APSMT, where  $\kappa$  is the error parameter. That is,  $\mathbf{R}$  should correctly receive  $m$  with probability at least  $(1 - 2^{-\Omega(\kappa)})$ . The probability is over the choice of  $m$  and the coin flips of all the nodes in  $\mathcal{N}$  and  $\mathcal{A}_{t_b}^{static}$ .

We now give an overview of our results which will be presented in this chapter. Before that, we give the following remark.

**Remark 16.3 (Concept of Phase in Asynchronous Network)** In synchronous networks, we assume that the protocols operate as a sequence of phases, where a phase is a send from  $\mathbf{S}$  to  $\mathbf{R}$  or vice-versa. This is a valid assumption, because there is an upper bound on the time delay of every wire, as the network is synchronous. However, in asynchronous network, there is no upper bound on the time delay. Hence, in asynchronous network, we cannot assume that the protocols operate in phases. Rather, we consider whether interaction is allowed between  $\mathbf{S}$  and  $\mathbf{R}$  or not. Accordingly, we consider two extreme settings, as discussed earlier in this section.

## 16.2 Overview of Our Results for PSMT and SSMT in Asynchronous Network

Our contributions in this chapter are as follows:

1. In [69], Sayeed et al. have given a PSMT protocol in asynchronous network, tolerating  $\mathcal{A}_{t_b}^{static}$  in the presence of  $n = 2t_b + 1$  unidirectional wires from  $\mathbf{S}$  to  $\mathbf{R}$ . However, we show that their protocol does not provide perfect security.
2. We show that if there are  $n$  unidirectional wires from  $\mathbf{S}$  to  $\mathbf{R}$  in an asynchronous network, then there exists a PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , iff  $n > 3t_b$ . Comparing this with Theorem 9.1, we find that synchrony of the network does *not* effect the possibility of PSMT protocol, if all the  $n$  wires are unidirectional from  $\mathbf{S}$  to  $\mathbf{R}$ .
3. We show that if there are  $n$  bi-directional wires between  $\mathbf{S}$  and  $\mathbf{R}$  in an asynchronous network, then there exists a PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , iff  $n > 3t_b$ . This is surprising because from Theorem 9.3,  $n > 2t_b$  bi-directional wires are necessary and sufficient for the existence of PSMT protocol against  $\mathcal{A}_{t_b}^{static}$  in synchronous network. This shows that if all the  $n$  wires are bi-directional, then synchrony of the network significantly affects the possibility of PSMT protocols.

4. We show that SSMT between  $\mathbf{S}$  and  $\mathbf{R}$  is possible in an asynchronous network, tolerating  $\mathcal{A}_{t_b}^{static}$  iff  $n > 2t_b$ . Moreover, this is true, irrespective of whether the  $n$  wires are unidirectional or the  $n$  wires are bi-directional. Comparing this with Theorem 14.1, we find that irrespective of whether the  $n$  wires are unidirectional or bi-directional, synchrony of the network does *not* affect the possibility of SSMT.

In [76], the authors have given the necessary and sufficient condition for PSMT and SSMT in asynchronous networks tolerating non-threshold adversary (see Section 1.4.3 for the meaning of non-threshold adversary). However, though not explicitly stated in the paper, their characterization for PSMT is true under the assumption that  $\mathbf{S}$  is honest, while their characterization for SSMT is true under the assumption that  $\mathbf{S}$  may be corrupted, where if  $\mathbf{S}$  is corrupted, then he may not send anything to  $\mathbf{R}$  along some path. Note that while in synchronous model,  $\mathbf{S}$  being honest or dishonest does not make any sense, in asynchronous model this makes lot of difference. This is because we cannot distinguish between a slow sender and a corrupted sender. However, in this chapter, we derive all the necessary and sufficient condition, assuming  $\mathbf{S}$  to be honest. The protocols given in [76] against non-threshold adversary are very complex. Though we can derive protocols for tolerating  $\mathcal{A}_{t_b}^{static}$  from the protocols of [76] tolerating non-threshold adversary, the resultant protocols will be very complex and inefficient. Instead, since we work on threshold model (where the corruption capability of the adversary is bounded by a threshold), our protocols are very elegant and efficient.

Asynchronous PSMT/SSMT is an important primitive for perfectly/statistically secure multiparty computation over asynchronous incomplete networks. Thus, our results can be used to transform the asynchronous secure computation protocols that run over a complete network [14, 12, 83, 64, 10] into ones that can be executed over incomplete networks.

In the next section, we discuss about APSMT, when all the wires are unidirectional from  $\mathbf{S}$  to  $\mathbf{R}$ .

### 16.3 APSMT When All Wires are Unidirectional from $\mathbf{S}$ to $\mathbf{R}$

In this section, we recall the existing APSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , proposed by Sayeed et al. [69] where all the  $n$  wires are directed from  $\mathbf{S}$  to  $\mathbf{R}$ . We show that their protocol does not provide perfect secrecy. We then give the true characterization for APSMT protocols tolerating  $\mathcal{A}_{t_b}^{static}$ , when all the  $n$  wires are unidirectional, directed from  $\mathbf{S}$  to  $\mathbf{R}$ .

#### 16.3.1 Existing APSMT Protocol of [69]

In [69], the authors have given an APSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by wires  $w_i, i = 1, \dots, n$ , directed from  $\mathbf{S}$  to  $\mathbf{R}$ , where  $n = 2t_b + 1$ . We briefly recall the protocol from [69] and show that the protocol does not achieve perfect secrecy; i.e.,  $\mathcal{A}_{t_b}^{static}$  can recover  $m^{\mathbf{S}}$ .

In the protocol, message  $m^{\mathbf{S}}$  belongs to the set  $Q = \{1, 2, \dots, m_{max}\}$  of positive integers, such that  $m_{max} > n$ . Let  $MAX = 2m_{max} + 1$ .  $\mathbf{S}$  sends  $m^{\mathbf{S}}$  by doing the following computation and communication:

1.  $\mathbf{S}$  randomly selects  $n$  values  $K_1, K_2, \dots, K_n$  from the set  $Q$  and associates  $K_i$  with wire  $w_i$ . For each  $K_i$ ,  $\mathbf{S}$  forms a *key-carrying-polynomial*  $p_i(x)$  of degree  $t_b$ , where  $p_i(0) = K_i$  and other coefficients of  $p_i(x)$  are randomly chosen from  $Q$ .  $\mathbf{S}$

also forms a *secret-carrying-polynomial*  $M(x)$  of degree  $n$ , where  $M(0) = m^{\mathbf{S}}$  and the coefficient of  $x^i$  is  $K_i$ .

2. Through wire  $w_i$ ,  $\mathbf{S}$  sends to  $\mathbf{R}$  the value  $p_j(i)$ , for  $j = 1, \dots, n$ .  $\mathbf{S}$  also broadcasts  $M(1)$  and  $M(MAX)$ , where the values of  $M(x)$  are from  $N$ , the infinite set of positive integers.

We now show how  $\mathcal{A}_{t_b}^{static}$  can recover  $m^{\mathbf{S}}$  from the values sent by  $\mathbf{S}$ . In the protocol,  $\mathbf{S}$  broadcasts:

$$\begin{aligned} V_1 &= M(1) = m^{\mathbf{S}} + K_1 + K_2 + \dots + K_n \text{ and} \\ V_2 &= M(MAX) = m^{\mathbf{S}} + (K_1 \times MAX) + (K_2 \times MAX^2) + \dots + (K_n \times MAX^n) \end{aligned}$$

Note that  $V_1$  and  $V_2$  does not belong to  $Q$ . They belong to  $N$ , the infinite set of positive integers; i.e., the protocol works with the exact values of  $V_1, V_2$ . However,  $m^{\mathbf{S}} \in Q$  and is always less than  $MAX$ . Since  $V_1$  and  $V_2$  are broadcasted,  $\mathcal{A}_{t_b}^{static}$  will also know  $V_1$  and  $V_2$ . Also  $MAX$  is a publicly known parameter. If  $\mathcal{A}_{t_b}^{static}$  computes  $(V_2 \bmod MAX)$ , then he obtains  $m^{\mathbf{S}}$ , because all other terms in  $V_2$  are multiple of  $MAX$ , except  $m^{\mathbf{S}}$ , which is less than  $MAX$ . Thus, protocol of [69] does not provide perfect secrecy. In fact, there does not exist any APSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  with  $n = 2t_b + 1$  unidirectional wires from  $\mathbf{S}$  to  $\mathbf{R}$ . In the sequel, we present the true characterization of APSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , when all the  $n$  wires are unidirectional from  $\mathbf{S}$  to  $\mathbf{R}$ .

### 16.3.2 Characterization of APSMT in Presence of Unidirectional Wires

We now prove the necessary and sufficiency condition for the existence of any APSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , when all the wires are directed from  $\mathbf{S}$  to  $\mathbf{R}$ .

**Theorem 16.4** *Suppose there exists  $n$  wires, directed from  $\mathbf{S}$  to  $\mathbf{R}$ , of which at most  $t_b$  could be under the control of  $\mathcal{A}_{t_b}^{static}$ . Then there exists an APSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , only if  $n > 3t_b$ .*

PROOF: From Theorem 9.1, we know that  $n > 3t_b$  wires are necessary for the existence of any synchronous PSMT protocol tolerating a  $t_b$ -active static Byzantine adversary, when all the wires are unidirectional from  $\mathbf{S}$  to  $\mathbf{R}$ . Hence it is obviously necessary for the existence of APSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , if all the wires are unidirectional from  $\mathbf{S}$  to  $\mathbf{R}$ .  $\square$

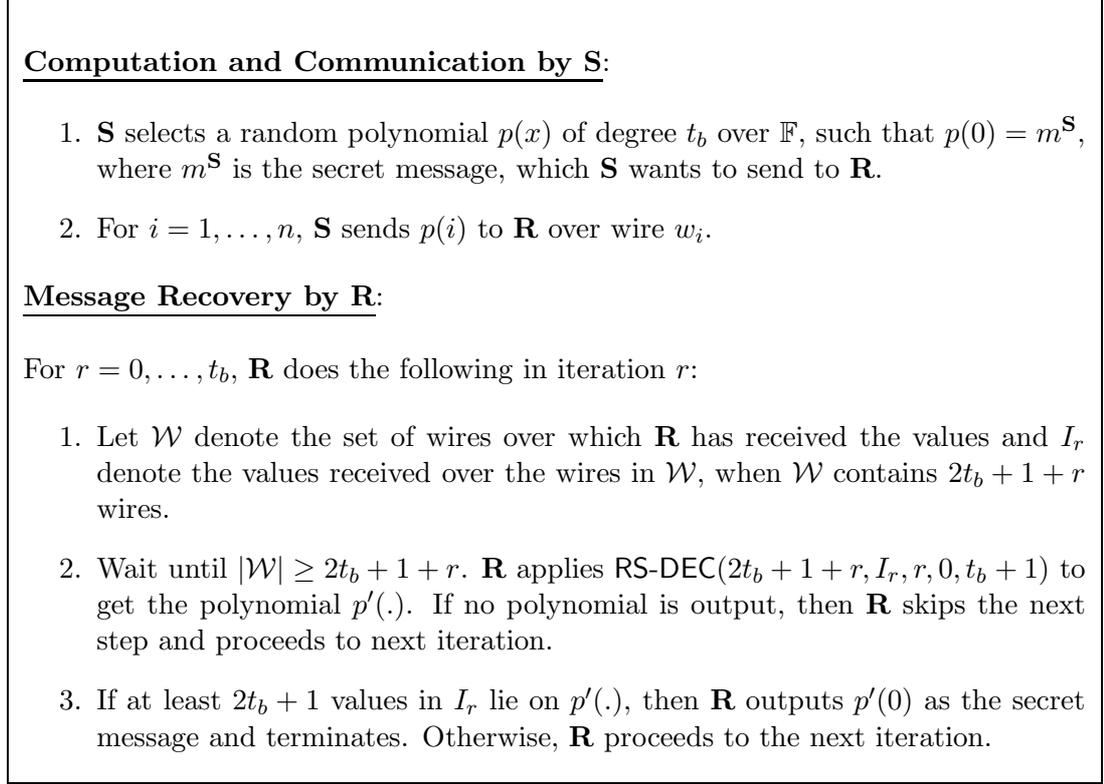
We now show that  $n > 3t_b$  unidirectional wires from  $\mathbf{S}$  to  $\mathbf{R}$  are also sufficient for designing an APSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ . For this, we design an APSMT protocol in the next section, tolerating  $\mathcal{A}_{t_b}^{static}$ , provided there exists  $n = 3t_b + 1$  unidirectional wires from  $\mathbf{S}$  to  $\mathbf{R}$ .

### 16.3.3 APSMT in Presence of $n = 3t_b + 1$ Unidirectional Wires

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by unidirectional wires  $w_i, 1 \leq i \leq n$ , which are directed from  $\mathbf{S}$  to  $\mathbf{R}$ , where  $n = 3t_b + 1$ . We design an APSMT protocol called  $\Pi_{APSMT}^{Unidirectional}$ , tolerating  $\mathcal{A}_{t_b}^{static}$ . The protocol is given in Fig. 16.1.

We now prove the properties of protocol  $\Pi_{APSMT}^{Unidirectional}$ .

Figure 16.1: Protocol  $\Pi_{APSM T}^{Unidirectional}$  with  $n = 3t_b + 1$  unidirectional wires from **S** to **R**.



**Theorem 16.5** *In protocol  $\Pi_{APSM T}^{Unidirectional}$ , the adversary  $\mathcal{A}_{t_b}^{static}$  gets no information about the secret message  $m^{\mathbf{S}}$ .*

PROOF: It is easy to see that  $\mathcal{A}_{t_b}^{static}$  gets at most  $t_b$  distinct points on  $p(x)$ . So  $\mathcal{A}_{t_b}^{static}$  lacks by one point to uniquely interpolate  $p(x)$ . This implies that  $p(0) = m^{\mathbf{S}}$  is information theoretically secure.  $\square$

**Theorem 16.6** *In protocol  $\Pi_{APSM T}^{Unidirectional}$ , **R** will eventually output  $m^{\mathbf{S}}$ .*

PROOF: Suppose  $\mathcal{A}_{t_b}^{static}$  corrupts  $\hat{r} \leq t_b$  wires during the transmission of values of  $p(x)$ . Now during  $\hat{r}^{th}$  iteration, **R** receives  $2t_b + 1 + \hat{r}$  points on  $p(x)$ , of which  $\hat{r}$  are corrupted. So from Theorem 2.19, RS-DEC will be able to correct the  $\hat{r}$  errors and hence the polynomial  $p'(\cdot)$  which is output by RS-DEC during  $\hat{r}^{th}$  iteration will pass through at least  $2t_b + 1$  points in  $I_r$ . Since out of these  $2t_b + 1$  points, at least  $t_b + 1$  are honest and uniquely define the original polynomial  $p(\cdot)$  ( $t_b + 1$  points uniquely define a  $t_b$  degree polynomial), the output polynomial  $p'(\cdot)$  is same as  $p(\cdot)$ . Thus  $p(\cdot)$  will be output in  $\hat{r}^{th}$  iteration and all the iterations up to iteration  $\hat{r}$  will be unsuccessful, as either they will not output any  $t_b$  degree polynomial or the output polynomial will not pass through  $2t_b + 1$  points in  $I_r$ <sup>1</sup>.  $\square$

**Theorem 16.7** *Let there exists  $n$  unidirectional wires from **S** to **R**. Then APSMT tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff  $n > 3t_b$ .*

PROOF: The proof follows from Theorem 16.4 and protocol  $\Pi_{APSM T}^{Unidirectional}$ .  $\square$

<sup>1</sup>The procedure used by **R** to recover the original polynomial is called **Online Error Correction (OEC)** [15], which is a well known technique in asynchronous settings. Informally, OEC allows **R** to find the original polynomial in an online fashion.

## 16.4 APSMT When All Wires are Bidirectional Between **S** and **R**

In this section, we characterize APSMT tolerating  $\mathcal{A}_{t_b}^{static}$ , when all the  $n$  wires between **S** and **R** are bi-directional. In this setting, we show that APSMT tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff there exists  $n > 3t_b$  bi-directional wires between **S** and **R**. This shows that irrespective of whether the  $n$  wires between **S** and **R** are uni-directional or bi-directional,  $n > 3t_b$  wires are necessary for the existence of any APSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ .

**Theorem 16.8** *Let **S** and **R** be connected by  $n = 3t_b + 1$  bi-directional wires, of which at most  $t_b$  are under the control of  $\mathcal{A}_{t_b}^{static}$ . Then there exists an APSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ .*

PROOF: Any bi-directional wire between **S** and **R** can be treated as an uni-directional wire from **S** to **R**. Now we know that there exists an APSMT protocol  $\Pi_{APSMT}^{Unidirectional}$  tolerating  $\mathcal{A}_{t_b}^{static}$  if there exists  $n = 3t_b + 1$  unidirectional wires from **S** to **R**. Hence the same protocol can also be executed if there exists  $n = 3t_b + 1$  bi-directional wires between **S** and **R**.  $\square$

We now show that if all the  $n$  wires between **S** and **R** are bi-directional, then APSMT tolerating  $\mathcal{A}_{t_b}^{static}$  is possible only if  $n > 3t_b$ . The proof is by contradiction. We first show that there does not exist any APSMT protocol, securely communicating a message between a sender **S'** and receiver **R'**, with three bi-directional wires between **S'** and **R'**, of which one can be corrupted by the adversary (Theorem 16.9). Then by using a standard player partitioning argument [44], we show that if there exists an APSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  with  $n = 3t_b$  bi-directional wires between **S** and **R**, then there exists an APSMT protocol tolerating  $\mathcal{A}_1^{static}$ , with 3 bi-directional wires between **S'** and **R'**, which is a contradiction (Theorem 16.10).

**Theorem 16.9** *Let there be three bi-directional wires between a sender **S'** and a receiver **R'**, of which at most one wire could be under the control of the adversary. Then there does not exist any APSMT protocol, securely transmitting a message from **S'** to **R'**.*

PROOF: The proof is by contradiction. Let **S'** and **R'** be connected by three bi-directional wires  $w_1, w_2, w_3$ , of which at most one wire can be under the control of adversary  $\mathcal{A}_1^{static}$ . Moreover, let  $\Pi$  be an APSMT protocol for securely transmitting a message from **S'** to **R'**, tolerating  $\mathcal{A}_1^{static}$ . Let  $E$  be an execution of  $\Pi$ . Then we define the following variables:

1.  $time(E, \mathbf{R}', w_i)$ : denotes the arrival time of the different messages (with respect to local clock) received by **R'** in  $E$ , along wire  $w_i, i \in \{1, 2, 3\}$ .
2.  $time(E, \mathbf{S}', w_i)$ : denotes the arrival time of the different messages (with respect to local clock) received by **S'** in  $E$ , along wire  $w_i, i \in \{1, 2, 3\}$ .
3.  $E^{time}$ : denotes the total time taken (with respect to **R'**) by execution  $E$ ; i.e., the time at which **R'** terminates by outputting the message in  $E$ .

From the termination property of APSMT, each execution of  $\Pi$  will eventually terminate. Moreover, in any execution of  $\Pi$ , the distribution of data sent along a single

wire will be same, irrespective of the secret message, which is sent during protocol  $\Pi$ . Otherwise, the adversary can passively listen the wire and will get information about the secret message, thus violating the perfect secrecy property of  $\Pi$ . Now consider the following execution sequences of protocol  $\Pi$ :

1.  $E_1$ : Let the random coin tosses of  $\mathbf{S}'$  and  $\mathbf{R}'$  be  $r_1$  and  $r_2$  respectively and let  $\mathbf{S}'$  wants to send the secret message  $m$ . The adversary strategy is to control wire  $w_3$  and not allowing any data to pass over  $w_3$  throughout  $E_1$ . Let  $\alpha$  and  $\beta$  denote the messages that are exchanged between  $\mathbf{S}'$  and  $\mathbf{R}'$ , along  $w_1$  and  $w_2$  respectively. The protocol terminates after working for  $E_1^{time}$  and outputs  $m$ .
2.  $E_2$ : Let the random coin tosses of  $\mathbf{S}'$  and  $\mathbf{R}'$  are  $r_1$  and  $r_2$  respectively and let  $\mathbf{S}'$  wants to send the secret message  $m$ . The adversary strategy is to passively control  $w_2$  and delay any information along  $w_3$  for time  $E_1^{time} + E_3^{time} + 1$  ( $E_3$  is defined below). In addition, the adversary schedules the messages along  $w_1$  and  $w_2$  in such a way that  $time(E_2, \mathbf{S}', w_i) = time(E_1, \mathbf{S}', w_i)$ , for  $i \in \{1, 2\}$  and  $time(E_2, \mathbf{R}', w_i) = time(E_1, \mathbf{R}', w_i)$ , for  $i \in \{1, 2\}$ . Thus the view of  $\mathbf{S}'$  and  $\mathbf{R}'$  in  $E_1$  and  $E_2$  are exactly same and hence the secret  $m$  is reconstructed. Also  $E_1^{time} = E_2^{time}$  and  $\alpha$  and  $\beta$  are exchanged between  $\mathbf{S}'$  and  $\mathbf{R}'$ , along  $w_1$  and  $w_2$  respectively.

Let  $m^* (\neq m)$  be another secret message. Then from the perfect secrecy property of  $\Pi$ , there exists  $r_3 (\neq r_1)$  and  $r_4 (\neq r_2)$ , such that the following holds:  $\mathbf{S}'$  wants to send  $m^*$ , the random coin tosses of  $\mathbf{S}'$  and  $\mathbf{R}'$  are  $r_3$  and  $r_4$  respectively and the information exchanged between  $\mathbf{S}'$  and  $\mathbf{R}'$  along wire  $w_2$  is  $\beta$ . Note that such an  $r_3, r_4$  exists, otherwise it implies that data sent along wire  $w_2$  is dependent on secret message, thus violating perfect secrecy property of  $\Pi$ . Now consider the following executions of  $\Pi$ :

3.  $E_3$ : Here, the random coin tosses of  $\mathbf{S}'$  and  $\mathbf{R}'$  are  $r_3$  and  $r_4$  respectively.  $\mathbf{S}'$  wants to send the secret message  $m^*$ . The adversary strategy is to control wire  $w_3$  and not allowing any data to pass over  $w_3$  throughout  $E_3$ . Let  $\alpha^*$  and  $\beta^* (= \beta)$  denote the messages that are exchanged between  $\mathbf{S}'$  and  $\mathbf{R}'$ , along  $w_1$  and  $w_2$  respectively. The protocol terminates after working for  $E_3^{time}$  and outputs  $m^*$ .
4.  $E_4$ : The random coin tosses of  $\mathbf{S}'$  and  $\mathbf{R}'$  are  $r_3$  and  $r_4$  respectively.  $\mathbf{S}'$  wants to send the secret message  $m^*$ . The adversary strategy is to passively control  $w_2$  and delay any information along  $w_3$  for time  $E_1^{time} + E_3^{time} + 1$ . In addition, the adversary schedules the messages along  $w_1$  and  $w_2$  in such a way that  $time(E_4, \mathbf{S}', w_i) = time(E_3, \mathbf{S}', w_i)$ , for  $i \in \{1, 2\}$  and  $time(E_4, \mathbf{R}', w_i) = time(E_3, \mathbf{R}', w_i)$ , for  $i \in \{1, 2\}$ . Thus the view of  $\mathbf{S}'$  and  $\mathbf{R}'$  in  $E_3$  and  $E_4$  are same and hence the secret  $m^*$  is reconstructed. Also  $E_3^{time} = E_4^{time}$  and  $\alpha^*$  and  $\beta^* (= \beta)$  are exchanged between  $\mathbf{S}'$  and  $\mathbf{R}'$ , along  $w_1$  and  $w_2$  respectively.
5.  $E_5$ : The random coin tosses of  $\mathbf{S}'$  and  $\mathbf{R}'$  are  $r_1$  and  $r_4$  respectively.  $\mathbf{S}'$  wants to send the secret message  $m$ . Let  $\alpha', \beta' (= \beta)$  denote the messages that should have been exchanged between  $\mathbf{S}'$  and  $\mathbf{R}'$  along  $w_1$  and  $w_2$  in ideal situation, when  $w_1$  and  $w_2$  are honest (not under the control of adversary).

Now the adversary strategy in  $E_5$  is as follows: adversary delay any information along  $w_3$  for time  $E_1^{time} + E_3^{time} + 1$ . In addition, the adversary controls  $w_1$  in Byzantine fashion, such that instead of receiving messages from  $\alpha'$ ,  $\mathbf{R}'$  gets messages from  $\alpha^*$ , while  $\mathbf{S}'$  receives messages from  $\alpha$ . Moreover, adversary schedules the messages along  $w_1$  and  $w_2$  in such a way that  $time(E_5, \mathbf{S}', w_i) =$

$time(E_2, \mathbf{S}', w_i)$ , for  $i \in \{1, 2\}$  and  $time(E_5, \mathbf{R}', w_i) = time(E_4, \mathbf{R}', w_i)$ , for  $i \in \{1, 2\}$ . Thus the view of  $\mathbf{S}'$  is  $\alpha \beta' = \alpha \beta$ , while view of  $\mathbf{R}'$  is  $\alpha^* \beta' = \alpha^* \beta$ .

Thus the view of  $\mathbf{S}'$  in  $E_2$  and  $E_5$  are same, so  $\mathbf{S}'$  will assume that  $m$  has been communicated securely. However, the view of  $\mathbf{R}'$  in  $E_5$  is same as in  $E_4$  and hence  $\mathbf{R}'$  will output  $m^*$ . But this violates the perfect reliability property of  $\Pi$ , which is a contradiction. Hence  $\Pi$  does not exist.  $\square$

**Theorem 16.10** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n$  bi-directional wires, of which at most  $t_b$  can be under the control of  $\mathcal{A}_{t_b}^{static}$ . Then there exists an APSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  only if  $n > 3t_b$ .*

PROOF: The proof is by contradiction. Assume that there exist an APSMT protocol  $\Pi^{APSMT}$  tolerating  $\mathcal{A}_{t_b}^{static}$ , where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 3t_b$  bi-directional wires. Now by using standard player partitioning strategy, we show how to transform protocol  $\Pi^{APSMT}$  into another APSMT protocol  $\Pi$ , which securely transmits a message from a sender  $\mathbf{S}'$  to a receiver  $\mathbf{R}'$ , who are connected by three bi-directional wires, of which at most one could be corrupted by the adversary. Let the wires between  $\mathbf{S}$  and  $\mathbf{R}$  be numbered  $1, 2, \dots, 3t_b$ . Similarly, let the wires between  $\mathbf{S}'$  and  $\mathbf{R}'$  be numbered as  $1, 2, 3$ . Now we define a mapping  $M : \{1 \dots n\} \longrightarrow \{1, 2, 3\}$  as follows:

$$\begin{aligned} M(x) &= 1 : \forall x \in \{1 \dots t_b\} \\ &= 2 : \forall x \in \{t_b + 1 \dots 2t_b\} \\ &= 3 : \forall x \in \{2t_b + 1 \dots 3t_b\} \end{aligned}$$

We denote  $M^{-1}(1) = \{1, 2, \dots, t_b\}$ ,  $M^{-1}(2) = \{t_b + 1, t_b + 2, \dots, 2t_b\}$  and  $M^{-1}(3) = \{2t_b + 1, 2t_b + 2, \dots, 3t_b\}$ . Now  $\Pi$  is obtained from  $\Pi^{APSMT}$  in the following way: if in protocol  $\Pi^{APSMT}$ ,  $k \in \mathbb{F}$  is sent from  $\mathbf{S}$  to  $\mathbf{R}$  on wire  $w \in \{1, 2, \dots, 3t_b\}$ , then in protocol  $\Pi$ ,  $k$  is sent from  $\mathbf{S}'$  to  $\mathbf{R}'$  on wire  $M(w)$ . We define the transmission from  $\mathbf{R}'$  to  $\mathbf{S}'$  in a similar fashion. Similarly, if the adversary controls wire  $w \in \{1, 2, 3\}$  in protocol  $\Pi$ , then he controls the set  $M^{-1}(w)$  in protocol  $\Pi^{APSMT}$ . It can be easily verified that the view of  $\mathbf{S}'$  and  $\mathbf{R}'$  in  $\Pi$  is same as the view of  $\mathbf{S}$  and  $\mathbf{R}$  respectively, in protocol  $\Pi^{APSMT}$ . So  $\Pi$  is an APSMT protocol, which securely transmits a message from  $\mathbf{S}'$  to  $\mathbf{R}'$ , who are connected by three bi-directional wires, of which at most one can be corrupted. But from Theorem 16.9,  $\Pi$  does not exist. Hence  $\Pi^{APSMT}$  also does not exist.  $\square$

## 16.5 ASSMT When All Wires are Unidirectional from $\mathbf{S}$ to $\mathbf{R}$

We now give the characterization for ASSMT protocols tolerating  $\mathcal{A}_{t_b}^{static}$ , when all the wires are directed from  $\mathbf{S}$  to  $\mathbf{R}$ .

**Theorem 16.11** *Let there exist  $n$  wires directed from  $\mathbf{S}$  to  $\mathbf{R}$ , of which at most  $t_b$  could be under the control of  $\mathcal{A}_{t_b}^{static}$ . Then there exists an ASSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , only if  $n > 2t_b$ .*

PROOF: From Theorem 14.1, we know that  $n > 2t_b$  wires are necessary for the existence of any synchronous SSMT protocol tolerating an all powerful  $t_b$ -active static Byzantine adversary, when all the wires are unidirectional from  $\mathbf{S}$  to  $\mathbf{R}$ . Hence it is obviously necessary for the existence of ASSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , if all the

wires are unidirectional from  $\mathbf{S}$  to  $\mathbf{R}$ .  $\square$

We now show that  $n = 2t_b + 1$  unidirectional wires from  $\mathbf{S}$  to  $\mathbf{R}$  are sufficient to design an ASSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ . Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + 1$  unidirectional wires, directed from  $\mathbf{S}$  to  $\mathbf{R}$ . Let the wires be denoted by  $w_1, \dots, w_n$ . Moreover, let  $\mathbb{F} = GF(2^\kappa)$ , where  $\kappa$  is the error parameter. Furthermore, without loss of generality, let  $n = \text{poly}(\kappa)$ . We now present an ASSMT protocol called  $\Pi_{ASSMT}^{Unidirectional}$ , which securely sends a message  $m^{\mathbf{S}} \in \mathbb{F}$ . The protocol is formally given in Fig. 16.2.

We now prove the properties of protocol  $\Pi_{ASSMT}^{Unidirectional}$ .

**Claim 16.12** *In protocol  $\Pi_{ASSMT}^{Unidirectional}$ , if  $\mathbf{R}$  concludes that  $p^{\mathbf{R}}(i)$  is a valid point, then except with error probability  $2^{-\Omega(\kappa)}$ ,  $p^{\mathbf{R}}(i) = p^{\mathbf{S}}(i)$ .*

PROOF: The claim trivially holds without any error if  $w_i$  is honest because an honest wire will correctly deliver  $p^{\mathbf{R}}(i) = p^{\mathbf{S}}(i)$ . So we consider the case when  $w_i$  is corrupted. So let  $w_i$  be a corrupted wire, who delivers  $p^{\mathbf{R}}(i) \neq p^{\mathbf{S}}(i)$ . In order that  $p^{\mathbf{R}}(i)$  is considered as a valid point, it must hold that  $\text{Support}_i \geq t_b + 1$ . This further implies that there exists at least one honest wire, say  $w_j$ , such that  $w_j \in \text{Support}_i$  because there can be at most  $t_b$  corrupted parties. Since  $w_j \in \text{Support}_i$ , it implies that  $\gamma_{ij}^{\mathbf{R}} = \text{Urauth}(p^{\mathbf{R}}(i); a_{ij}^{\mathbf{R}}, b_{ij}^{\mathbf{R}})$ . Now notice that  $w_j$  is an honest wire and so  $a_{ij}^{\mathbf{R}} = a_{ij}^{\mathbf{S}}$  and  $b_{ij}^{\mathbf{R}} = b_{ij}^{\mathbf{S}}$ . However  $\mathcal{A}_{t_b}^{static}$  will have no information about  $a_{ij}^{\mathbf{R}}$  and  $b_{ij}^{\mathbf{R}}$ , as they are sent over  $w_j$ . So from the properties of  $\text{Urauth}$ , except with probability  $2^{-\Omega(\kappa)}$ ,  $\gamma_{ij}^{\mathbf{R}} \neq \text{auth}(p^{\mathbf{R}}(i); a_{ij}^{\mathbf{R}}, b_{ij}^{\mathbf{R}})$ , which is a contradiction. So except with error probability  $2^{-\Omega(\kappa)}$ ,  $p^{\mathbf{R}}(i) = p^{\mathbf{S}}(i)$ .  $\square$

**Claim 16.13** *In protocol  $\Pi_{ASSMT}^{Unidirectional}$ ,  $\mathbf{R}$  will eventually get  $t_b + 1$  valid points.*

PROOF: In  $\Pi_{ASSMT}^{Unidirectional}$ , the worst case occurs when at most  $t_b$  corrupted wires do not deliver any information at all. However, still there exists  $t_b + 1$  honest wires, who will eventually deliver correct points to  $\mathbf{R}$ . These correct points will eventually reach  $\mathbf{R}$  and hence will be considered as valid points by  $\mathbf{R}$ .  $\square$

**Claim 16.14** *In protocol  $\Pi_{ASSMT}^{Unidirectional}$ , if  $\mathbf{R}$  outputs  $m^{\mathbf{R}}$ , then except with probability  $2^{-\Omega(\kappa)}$ ,  $m^{\mathbf{R}} = m^{\mathbf{S}}$ .*

PROOF: If  $\mathbf{R}$  outputs  $m^{\mathbf{R}}$ , then it implies that  $\mathbf{R}$  must have received  $t_b + 1$  valid points, using which  $\mathbf{R}$  has interpolated  $t_b$  degree polynomial  $p^{\mathbf{R}}(x)$ , such that  $p^{\mathbf{R}}(0) = m^{\mathbf{R}}$ . In the worst case, out of these  $t_b + 1$  valid points, at most  $t_b$  points could have been received over the wires which are under the control of  $\mathcal{A}_{t_b}^{static}$ . However, from Claim 16.12, the probability that none of those  $t_b$  points are the original points on  $p^{\mathbf{S}}(x)$  is at most  $t_b 2^{-\Omega(\kappa)} \approx 2^{-\Omega(k)}$ . So except with probability  $2^{-\Omega(k)}$ , all the  $t_b + 1$  valid points are indeed the original points on  $p^{\mathbf{S}}(x)$ . So  $m^{\mathbf{R}} = m^{\mathbf{S}}$ , except with probability  $2^{-\Omega(\kappa)}$ .  $\square$

**Claim 16.15** *In protocol  $\Pi_{ASSMT}^{Unidirectional}$ ,  $\mathcal{A}_{t_b}^{static}$  will get no information about  $m^{\mathbf{S}}$ .*

PROOF: Without loss of generality, let  $w_1, \dots, w_{t_b}$  be under the control of  $\mathcal{A}_{t_b}^{static}$ . So  $\mathcal{A}_{t_b}^{static}$  will know  $p^{\mathbf{S}}(1), \dots, p^{\mathbf{S}}(t_b)$ .  $\mathcal{A}_{t_b}^{static}$  will also know the authentication keys  $(a_{ji}^{\mathbf{S}}, b_{ji}^{\mathbf{S}})$ , for  $j = 1, \dots, n$  and  $i = 1, \dots, t_b$ . But since the authentication keys used to authenticate each point on  $p^{\mathbf{S}}(x)$  are completely random and independent of each other, they do not provide any extra information to  $\mathcal{A}_{t_b}^{static}$  about  $p^{\mathbf{S}}(t_b + 1), \dots, p^{\mathbf{S}}(n)$ . Thus adversary will lack by one point to uniquely interpolate  $p^{\mathbf{S}}(x)$  and so  $p^{\mathbf{S}}(0) = m^{\mathbf{S}}$  will be information theoretically secure.  $\square$

Figure 16.2: Protocol  $\Pi_{ASSMT}^{Unidirectional}$  with  $n = 2t_b + 1$  unidirectional wires from  $\mathbf{S}$  to  $\mathbf{R}$ .

**Computation and Communication by  $\mathbf{S}$ :**

1.  $\mathbf{S}$  selects a random polynomial  $p^{\mathbf{S}}(x)$  of degree  $t_b$  over  $\mathbb{F}$ , such that  $p^{\mathbf{S}}(0) = m^{\mathbf{S}}$ , where  $m^{\mathbf{S}}$  is the secret message, which  $\mathbf{S}$  wants to send to  $\mathbf{R}$ .
2. For  $i = 1, \dots, n$ ,  $\mathbf{S}$  computes  $p^{\mathbf{S}}(i)$ .
3. For  $i = 1, \dots, n$ , corresponding to  $p^{\mathbf{S}}(i)$ ,  $\mathbf{S}$  randomly selects  $n$  non-zero authentication keys  $(a_{ij}^{\mathbf{S}}, b_{ij}^{\mathbf{S}}) \in \mathbb{F}^2$ , for  $j = 1, \dots, n$ .
4. For  $i = 1, \dots, n$ ,  $\mathbf{S}$  computes  $\gamma_{ij}^{\mathbf{S}} = URauth(p^{\mathbf{S}}(i); a_{ij}^{\mathbf{S}}, b_{ij}^{\mathbf{S}})$ , for  $j = 1, \dots, n$ .
5. For  $i = 1, \dots, n$ ,  $\mathbf{S}$  sends the following to  $\mathbf{R}$  over wire  $w_i$ :
  - (a) The value  $p^{\mathbf{S}}(i)$ ;
  - (b)  $\gamma_{ij}^{\mathbf{S}}$ , for  $j = 1, \dots, n$ ;
  - (c) The authentication keys  $(a_{ji}^{\mathbf{S}}, b_{ji}^{\mathbf{S}})$ , for  $j = 1, \dots, n$ .

**Message Recovery by  $\mathbf{R}$ :**

For  $r = 0, \dots, t_b$ ,  $\mathbf{R}$  does the following in iteration  $r$ :

1. Let  $\mathcal{W}$  denote the set of wires  $w_i$  over which  $\mathbf{R}$  has received a complete set of values; i.e.,
  - (a) The value  $p^{\mathbf{R}}(i)$ ;
  - (b)  $\gamma_{ij}^{\mathbf{R}}$ , for  $j = 1, \dots, n$ ;
  - (c) The authentication keys  $(a_{ji}^{\mathbf{R}}, b_{ji}^{\mathbf{R}})$ , for  $j = 1, \dots, n$ .

Let  $W_r$  denote the contents of  $\mathcal{W}$ , when  $\mathcal{W}$  contains exactly  $t_b + 1 + r$  wires.

2. Wait until  $|\mathcal{W}| \geq t_b + 1 + r$ . Now corresponding to every  $w_i \in W_r$ ,  $\mathbf{R}$  computes

$$Support_i = \{w_j \in W_r : \gamma_{ij}^{\mathbf{R}} = URauth(p^{\mathbf{R}}(i); a_{ij}^{\mathbf{R}}, b_{ij}^{\mathbf{R}})\}$$

3. If  $Support_i \geq t_b + 1$ , then  $\mathbf{R}$  concludes that  $p^{\mathbf{R}}(i)$  is a valid point.
4. If  $\mathbf{R}$  finds  $t_b + 1$  valid points, then using them  $\mathbf{R}$  interpolates the  $t_b$  degree polynomial  $p^{\mathbf{R}}(x)$ , outputs  $m^{\mathbf{R}} = p^{\mathbf{R}}(0)$  and terminates the protocol. Otherwise  $\mathbf{R}$  proceeds to the next iteration.

**Theorem 16.16** *If there are  $n = 2t_b + 1$  unidirectional wires from  $\mathbf{S}$  to  $\mathbf{R}$ , then there exists an efficient ASSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ .*

PROOF: Follows from protocol  $\Pi_{ASSMT}^{Unidirectional}$  and Claim 16.12, Claim 16.13, Claim 16.14 and Claim 16.15.  $\square$

**Theorem 16.17** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n$  unidirectional wires, directed from  $\mathbf{S}$  to  $\mathbf{R}$ . Then ASSMT tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff  $n > 2t_b$ .*

PROOF: Follows from Theorem 16.11 and Theorem 16.16.  $\square$

## 16.6 ASSMT When All Wires are Bidirectional Between $\mathbf{S}$ to $\mathbf{R}$

The characterization for ASSMT tolerating  $\mathcal{A}_{t_b}^{static}$ , when all the  $n$  wires between  $\mathbf{S}$  and  $\mathbf{R}$  are bi-directional is given by following theorem:

**Theorem 16.18** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n$  bi-directional wires, of which at most  $t_b$  could be under the control of  $\mathcal{A}_{t_b}^{static}$ . Then there exists an ASSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$  iff  $n > 2t_b$ .*

PROOF: Any bi-directional wire between  $\mathbf{S}$  and  $\mathbf{R}$  can be treated as an uni-directional wire from  $\mathbf{S}$  to  $\mathbf{R}$ . Now we know that there exists an ASSMT protocol  $\Pi_{ASSMT}^{Unidirectional}$  tolerating  $\mathcal{A}_{t_b}^{static}$  if there exists  $n = 2t_b + 1$  unidirectional wires from  $\mathbf{S}$  to  $\mathbf{R}$ . Hence the same protocol can also be executed if there exists  $n = 2t_b + 1$  bi-directional wires between  $\mathbf{S}$  to  $\mathbf{R}$ . This proves the sufficiency part.

From Theorem 14.1, we know that  $n > 2t_b$  wires are necessary for the existence of any synchronous SSMT protocol tolerating an all powerful  $t_b$ -active Byzantine adversary, when all the wires are bi-directional. Hence it is obviously necessary for the existence of ASSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ , if all the wires between  $\mathbf{S}$  and  $\mathbf{R}$  bi-directional.  $\square$

## 16.7 Concluding Remarks and Open Problems

In this chapter, we have studied PSMT and SSMT in asynchronous networks. We showed that the existing PSMT protocol of [70] does not provide perfect secrecy. We have then given the exact characterization of PSMT in asynchronous networks. We have also given the necessary and sufficient condition for SSMT in asynchronous networks. Our characterization reveals that asynchrony of the network demands higher connectivity requirement for PSMT. On the other hand, asynchrony of the network does not demand higher connectivity requirement for SSMT. Our results are summarized in Fig. 16.3.

Figure 16.3: Connectivity Requirement for PSMT and SSMT in Asynchronous Network Tolerating  $\mathcal{A}_{t_b}^{static}$

Type of Wires	PSMT	SSMT
Unidirectional	$n \geq 3t_b + 1$	$n \geq 2t_b + 1$
Bidirectional	$n \geq 3t_b + 1$	$n \geq 2t_b + 1$

In this chapter, we have considered two extreme cases: when all the wires are uni-directional and when all the wires are bidirectional. Moreover, we have only considered only Byzantine corruption. This brings forth the following open problems:

**Open Problem 19** *What is the necessary and sufficient condition for PSMT and SSMT in asynchronous network tolerating  $\mathcal{A}_{t_b}^{static}$ , when certain wires are directed from  $\mathbf{S}$  to  $\mathbf{R}$  and certain wires are directed from  $\mathbf{R}$  to  $\mathbf{S}$ ?*

**Open Problem 20** *What is the necessary and sufficient condition for PSMT and SSMT in asynchronous network tolerating mixed adversary.*

## Chapter 17

# Conclusion and Directions for Further Research

In this chapter, we summarize our contribution in this thesis. We also mention several problems for future directions.

### 17.1 Summary of Our Contributions

In this dissertation, we looked into the issues of POSSIBILITY, FEASIBILITY and OPTIMALITY of RMT/SMT and its variants in several network models and adversarial settings. This thesis reports several new improved/efficient/optimal solutions, gives affirmative/negative answers to several significant open problems and last but not the least, provides first solutions to several newly formulated problems. We now briefly summarize our main achievements of this thesis:

- We designed a *three* phase communication optimal PRMT protocol in undirected synchronous network, tolerating  $\mathcal{A}_{t_b}^{static}$ . This significantly improves the previous best known communication optimal PRMT protocol in undirected synchronous network, tolerating  $\mathcal{A}_{t_b}^{static}$ , which takes  $\mathcal{O}(\log t_b)$  phases.
- We designed a *three* phase communication optimal PRMT protocol tolerating  $\mathcal{A}_{t_b}^{mobile}$ , which sends a sufficiently large message containing  $\ell$  field elements by communicating  $\mathcal{O}(\ell)$  field elements. This is the first protocol of its type in the literature. We also derived a tight bound on the number of communication rounds required to achieve reliable communication from **S** to **R** tolerating a mobile adversary with arbitrary *roaming speed*.
- We studied the inherent tradeoff among the three important parameters of PRMT, namely the connectivity requirement  $n$ , phase complexity  $r$  and communication complexity  $b$ , in the presence of a static mixed adversary  $\mathcal{A}_{(t_b, t_f)}^{static}$ . Specifically, we resolved the *Holy Grail* problem of PRMT. Our lower bound is first of its kind and captures the inherent tradeoff among  $n, b, \ell$  and  $r$  simultaneously.
- We resolved the issue of POSSIBILITY, FEASIBILITY and OPTIMALITY of PRMT in undirected synchronous network, tolerating mobile mixed adversary  $\mathcal{A}_{(t_b, t_f)}^{mobile}$ , which is done for the first time in the literature. Our results show the following interesting fact: irrespective of whether the adversary is static or mobile, the necessary and sufficient condition for PRMT is same. However, if the adversary

is mixed, then any PRMT protocol against mobile adversary requires more communication, as compared to its static counter part. Thus if the adversary can do *mixed* type of corruption, then mobility of the adversary affects OPTIMALITY of the protocols.

- We resolved the issue of OPTIMALITY of PRMT in directed synchronous network tolerating  $\mathcal{A}_{t_b}^{static}$ , which is done for the first time in the literature.
- We presented an efficient, three phase, communication optimal PSMT protocol in undirected synchronous network, tolerating  $\mathcal{A}_{t_b}^{static}$ .
- We designed a *three* phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{t_b}^{mobile}$ . Our communication optimal PSMT protocol against  $\mathcal{A}_{t_b}^{mobile}$  gives the following conclusion: if the adversary does only Byzantine corruption, then mobility of the adversary does not hinder to design a three phase PSMT with a communication complexity of  $\mathcal{O}(n\ell)$ .
- We completely resolved the issue of OPTIMALITY of multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . Specifically, we presented a four phase communication optimal PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , which securely sends a message containing  $\ell$  field elements, by communicating  $\mathcal{O}(n\ell)$  field elements over  $n = 2t_b + t_f + t_p + 1$  wires.
- We completely resolved the issue of POSSIBILITY and OPTIMALITY of PSMT in undirected synchronous network, tolerating mobile mixed adversary  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .
- We provided communication optimal PSMT protocols in directed synchronous network tolerating  $\mathcal{A}_{t_b}^{static}$ , which are first of their own kind.
- We completely resolved the issue of OPTIMALITY of multi phase SSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .
- We showed that the existing SSMT protocols of [24, 87, 54] in directed synchronous network, tolerating  $\mathcal{A}_{t_b}^{static}$  are *inefficient*. We then presented new and efficient, communication optimal SSMT protocol in directed synchronous network, tolerating  $\mathcal{A}_{t_b}^{static}$ .
- We showed that the existing PSMT protocol of [69] in asynchronous network does not provide perfect security. We then give the characterization for PSMT and SSMT in asynchronous network tolerating  $\mathcal{A}_{t_b}^{static}$ , thus completely resolving the issue of POSSIBILITY. The most interesting fact brought forth by our characterization is the following: our characterization shows that *asynchrony of the network demands higher connectivity of the network for the existence of PSMT protocols*. On the other hand, *asynchrony of the network does not demand higher connectivity of the network for SSMT protocols*.

## 17.2 Major Inferences Drawn From Our Results

Some of the major conclusions that we can draw from our studies on RMT/SMT are as follows:

1. In a minimally connected network tolerating  $\mathcal{A}_{t_f}^{static}$ , any PRMT protocol requires  $\Omega(\log(t_f))$  phases to achieve reliability with constant factor overhead. This is quiet interesting because against  $\mathcal{A}_{t_b}^{static}$ , reliability with constant factor overhead can be achieved in *three* phases.

2. Intuitively, one may feel that the connectivity requirement and communication complexity of PRMT/PSMT protocols should be more for tolerating  $\mathcal{A}_{t_b}^{mobile}$ , in comparison to  $\mathcal{A}_{t_b}^{static}$ . However surprisingly, we have shown that in an undirected synchronous network, the network connectivity and communication complexity of PRMT/PSMT protocol is same, irrespective of whether the adversary is  $\mathcal{A}_{t_b}^{static}$  or  $\mathcal{A}_{t_b}^{mobile}$ .

However, if the adversary can do mixed corruption, then any PRMT protocol against mobile adversary requires more communication, as compared to its static counterpart. Thus if the adversary can do *mixed* type of corruption, then mobility of the adversary affects OPTIMALITY of RMT protocols.

3. In undirected synchronous network, the network connectivity and communication complexity of *all* multi-phase PSMT protocols is same against  $\mathcal{A}_{t_b}^{static}$ . However, this is not the case in directed synchronous network. That is, the network connectivity and communication complexity of two phase PSMT in directed synchronous network is different from the one required for three or more phase PSMT in directed network tolerating  $\mathcal{A}_{t_b}^{static}$ .
4. Our studies show that asynchrony of the network demands higher network connectivity for the existence of PSMT protocols. On the other hand, asynchrony of the network does not affect the network connectivity requirement for SSMT protocols.

### 17.3 Directions for Future Research

Throughout the thesis, several open problems have been mentioned in each chapter. These problems are open in the context of this thesis. We now provide some future directions that are beyond the scope of this thesis.

1. **Arbitrary Directed Networks:** In this thesis, while considering the directed network model, we have abstracted the network as directed wires, which are directed either from **S** to **R** or vice-versa. However, as mentioned in Section 15.1.1, such an abstraction is valid if it is assumed that the intermediate nodes are only message forwarding nodes and do no other computation. However, in an *arbitrary directed* network, if the intermediate nodes (other than **S** and **R**) are allowed to carry out computation and communication (beyond just acting as a message forwarding node), as in the case of a *virtual private network (VPN)*, then the wired abstraction results in loss of generality. The insufficiency of wired abstraction in such a network model is pointed out in [84, 72, 80] where characterizations for SRMT and SSMT over the arbitrary network, treating entire graph in its full form are also reported. However, it is likely to take exponential time to verify whether a given arbitrary directed network satisfies the characterization given in [84, 72, 80] for the possibility of SRMT and SSMT. Moreover, the protocols given in [84, 72, 80] require exponential computational and communication complexity and are highly non-intuitive. Hence, it is an interesting open question to come up with efficient RMT/SMT protocols in arbitrary directed network, considering the network in its full form.
2. **Hypergraph Network Model:** In this thesis, we have only considered undirected and directed network model, which models point to point communication. However, in many scenarios, private one-to-one channels may not exist. Typical

examples include Radio transmission and LAN network. Also in many practical scenarios, a base station can broadcast to a set of receivers, but the other way around communication might not be possible. In these cases, directed hypergraph is the only way to model the network. Directed hypergraph is the most generic network model with the facility of multicasting. Unfortunately, not too much is known regarding the **POSSIBILITY**, **FEASIBILITY** and **OPTIMALITY** of RMT/SMT and its variants in hypergraph network model (see [34, 79, 87] for partial results). So it is important to pursue research in this direction.

**3. Non-Threshold Adversary Settings:** In this thesis, we have considered threshold adversary settings, where the corruption capacity of the adversary is bounded by a threshold. Modelling the adversary by a threshold helps in easy characterization of RMT/SMT. It also helps in analyzing protocols and proving lower bound on the communication complexity. However, as mentioned in [41], modelling the (dis)trust in the network as a threshold adversary is not always appropriate. This is due to the following reasons:

- (a) In the case of secure communication, not all scenarios of mutual (dis)trust can be captured by a threshold adversary.
- (b) The threshold model may lead to a gross overestimation of the connectivity requirement of the underlying network.

Motivated by this, the authors in [41] have studied PRMT and PSMT in non-threshold adversary settings. Specifically, the authors have resolved the issue of **POSSIBILITY** of PRMT and PSMT in undirected synchronous network, tolerating non-threshold static Byzantine adversary. The work of Kumar et al. was followed by [25, 84, 62, 79, 80], who gave the characterization of different variants of RMT/SMT in different network models, tolerating non-threshold adversary. However, to the best of our knowledge, nothing is known in the literature regarding **OPTIMALITY** of RMT/SMT and its variants in non-threshold model. Hence, it is an interesting open problem to look into the issue of **OPTIMALITY** of RMT/SMT and its variants in various network models in non-threshold settings.

**4. Incomplete Information About Network Topology:** All the existing results for RMT/SMT and all the results that are discussed in this thesis are derived under the assumption that the entire network topology is known to all the nodes in the network. A more realistic model is the one where each node may know *only* about his neighbors in the network up to some *constant* number of levels. Note that none of the protocols described in this thesis works correctly (in the worst case) in the absence of the full topological information. We remark that the design of RMT/SMT protocols with incomplete topological information (like each node knowing the identity of his neighbors alone) is several times more challenging and complex as compared to the case, when each node has full information about the topology of the network.

## List of Publications Related to the Thesis (in Reverse Chronological Order)

1. A. Patra, A. Choudhary, and C. Pandu Rangan. On communication complexity of secure message transmission in directed networks. In K. Kant, S. V. Premmaraju, K. M. Sivalingam, and J. Wu, editors, *Distributed Computing and Networking, 11th International Conference, ICDCN 2010, Kolkata, India, January 3 - 6, 2010, Proceedings*, volume 5935 of *Lecture Notes in Computer Science*, pages 42–53. Springer Verlag, 2010.
2. A. Patra, A. Choudhary, and C. Pandu Rangan. Brief announcement: Perfectly secure message transmission in directed networks revisited. In S. Tirthapura and L. Alvisi, editors, *Proceedings of the 28th Annual ACM Symposium on Principles of Distributed Computing, PODC 2009, Calgary, Alberta, Canada, August 10-12, 2009*, pages 278–279. ACM, 2009.
3. A. Choudhary, A. Patra, Ashwinkumar B. V, K. Srinathan, and C. Pandu Rangan. On minimal connectivity requirement for secure message transmission in asynchronous networks. In V. Garg, R. Wattenhofer, and K. Kothapalli, editors, *Distributed Computing and Networking, 10th International Conference, ICDCN 2009, Hyderabad, India, January 03-06, 2009*, volume 5408 of *Lecture Notes in Computer Science*, pages 148–162, 2009. Full version communicated to *Journal of Parallel and Distributed Computing*.
4. A. Patra, A. Choudhary, M. Vaidyanathan, and C. Pandu Rangan. Efficient perfectly reliable and secure message transmission tolerating mobile adversary. In Y. Mu, W. Susilo, and J. Seberry, editors, *Information Security and Privacy, 13th Australasian Conference, ACISP 2008, Wollongong, Australia, July 7-9, 2008, Proceedings*, volume 5107 of *Lecture Notes in Computer Science*, pages 170–186. Springer, 2008. Full version appeared in *International Journal of Applied Cryptography (IJACT)*: 1(3), pages 200-224, 2009.
5. A. Choudhary, A. Patra, B. V. Ashwinkumar, K. Srinathan, and C. Pandu Rangan. Perfectly Reliable and Secure Communication Tolerating Static and Mobile Mixed Adversary. In R. Safavi-Naini, editor, *Information Theoretic Security, Third International Conference, ICITS 2008, Calgary, Canada, August 10-13, 2008, Proceedings*, volume 5155 of *Lecture Notes in Computer Science*, pages 137–155. Springer, 2008. Full version communicated to *Theoretical Computer Science Journal*.
6. B. V. Ashwinkumar, A. Patra, A. Choudhary, K. Srinathan, and C. Pandu Rangan. On tradeoff between network connectivity, phase complexity and communication complexity of reliable communication tolerating mixed adversary. In R. A. Bazzi and B. Patt-Shamir, editors, *Proceedings of the Twenty-Seventh Annual ACM Symposium on Principles of Distributed Computing, PODC 2008, Toronto, Canada, August 18-21, 2008*, pages 115–124. ACM, 2008. Full version communicated to *Journal of ACM*.
7. K. Srinathan, A. Patra, A. Choudhary, and C. Pandu Rangan. Efficient single phase unconditionally secure message transmission with optimum communication complexity. In R. A. Bazzi and B. Patt-Shamir, editors, *Proceedings of the Twenty-Seventh Annual ACM Symposium on Principles of Distributed Computing, PODC 2008, Toronto, Canada, August 18-21, 2008*, pages 457. ACM, 2008.

8. A. Patra, A. Choudhury, and C. Pandu Rangan. Statistically reliable and secure message transmission in directed networks. Cryptology ePrint Archive, Report 2008/262. A preliminary version appeared in R. Ostrovsky, R. De Prisco, and I. Visconti, editors, *Security and Cryptography for Networks, 6th International Conference, SCN 2008, Amalfi, Italy, September 10-12, 2008. Proceedings*, volume 5229 of *Lecture Notes in Computer Science*, pages 309–326. Springer, 2008. Full version communicated to *Information and Computation Journal*.
9. A. Patra, B. Shankar, A. Choudhary, K. Srinathan, and C. Pandu Rangan. Perfectly secure message transmission in directed networks tolerating threshold and non threshold adversary. In F. Bao, S. Ling, T. Okamoto, H. Wang, and C. Xing, editors, *Cryptology and Network Security, 6th International Conference, CANS 2007, Singapore, December 8-10, 2007, Proceedings*, volume 4856 of *Lecture Notes in Computer Science*, pages 80–101. Springer, 2007.
10. A. Patra, A. Choudhary, K. Srinathan, and C. Pandu Rangan. Perfectly Reliable and Secure Communication in Directed Networks Tolerating Mixed Adversary. In A. Pelc, editors, *Distributed Computing, 21st International Symposium, DISC 2007, Lemesos, Cyprus, September 24-26, 2007, Proceedings*, volume 4731 of *Lecture Notes in Computer Science*, pages 496-498. Springer, 2007.
11. K. Srinathan, A. Patra, A. Choudhary, and C. Pandu Rangan. Probabilistic perfectly reliable and secure message transmission - possibility, feasibility and optimality. In K. Srinathan, C. Pandu Rangan, and M. Yung, editors, *Progress in Cryptology - INDOCRYPT 2007, 8th International Conference on Cryptology in India, Chennai, India, December 9-13, 2007, Proceedings*, volume 4859 of *Lecture Notes in Computer Science*, pages 101–122. Springer, 2007. Full version to appear in *International Journal of Applied Cryptography (IJACT)*.
12. A. Patra, A. Choudhary, and C. Pandu Rangan. Constant phase efficient protocols for secure message transmission in directed networks. In I. Gupta and R. Wattenhofer, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing, PODC 2007, Portland, Oregon, USA, August 12-15, 2007*, pages 322–323. ACM, 2007.
13. A. Patra, A. Choudhary, K. Srinathan, and C. Pandu Rangan. Constant phase bit optimal protocols for perfectly reliable and secure message transmission. In R. Barua and T. Lange, editors, *Progress in Cryptology - INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11-13, 2006, Proceedings*, volume 4329 of *Lecture Notes in Computer Science*, pages 221–235. Springer, 2006.

# Bibliography

- [1] Clique (graph theory). Wikipedia article.
- [2] Maximal independent set. Wikipedia article.
- [3] Tuřan graph. Wikipedia article.
- [4] S. Agarwal, R. Cramer, and R. de Haan. Asymptotically optimal two round perfectly secure message transmission. In C. Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 394–408. Springer-Verlag, 2006.
- [5] T. Araki. Almost secure 1-round message transmission scheme with polynomial time message decryption. In Reihaneh Safavi-Naini, editor, *Information Theoretic Security, Third International Conference, ICITS 2008, Calgary, Canada, August 10-13, 2008, Proceedings*, volume 5155 of *Lecture Notes in Computer Science*, pages 2–13. Springer, 2008.
- [6] B. V. Ashwinkumar, A. Patra, A. Choudhary, K. Srinathan, and C. Pandu Rangan. On tradeoff between network connectivity, phase complexity and communication complexity of reliable communication tolerating mixed adversary. In R. A. Bazzi and B. Patt-Shamir, editors, *Proceedings of the Twenty-Seventh Annual ACM Symposium on Principles of Distributed Computing, PODC 2008, Toronto, Canada, August 18-21, 2008*, pages 115–124. ACM, 2008.
- [7] M. Backes, C. Cachin, and R. Strobl. Proactive secure message transmission in asynchronous networks. In *PODC 2003, Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing (PODC 2003), July 13-16, 2003, Boston, Massachusetts, USA*, pages 223–232. ACM Press, 2003.
- [8] D. Beaver. Efficient multiparty protocols using circuit randomization. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432, 1992.
- [9] Z. Beerliová-Trubíniová and M. Hirt. Efficient Multi-party Computation with Dispute Control. In S. Halevi and T. Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 305–328. Springer, 2006.
- [10] Z. Beerliová-Trubíniová and M. Hirt. Simple and efficient perfectly-secure asynchronous MPC. In K. Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology*

- and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings, volume 4833 of *Lecture Notes in Computer Science*, pages 376–392. Springer Verlag, 2007.
- [11] Z. Beerliová-Trubíniová and M. Hirt. Perfectly secure MPC with linear communication complexity. In R. Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 213–230. Springer, 2008.
- [12] M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous secure computation. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, 1993*, pages 52–61. ACM Press, 1993.
- [13] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, 2-4 May 1988, Chicago, Illinois, USA*, pages 1–10. ACM, 1988.
- [14] M. BenOr, B. Kelmer, and T. Rabin. Asynchronous secure computations with optimal resilience. In *Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing, Los Angeles, California, USA, August 14-17*, pages 183–192. ACM Press, 1994.
- [15] R. Canetti. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, Weizmann Institute, Israel, 1995.
- [16] R. Canetti and T. Rabin. Fast Asynchronous Byzantine Agreement with Optimal Resilience. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, 1993*, pages 42–51. ACM, 1993.
- [17] D. Chaum, C. Crépeau, and I. Damgård. Multiparty Unconditionally Secure Protocols (extended abstract). In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, 2-4 May 1988, Chicago, Illinois, USA*, pages 11–19. ACM, 1988.
- [18] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). In *26th Annual Symposium on Foundations of Computer Science, 21-23 October 1985, Portland, Oregon, USA*, pages 383–395. IEEE, 1985.
- [19] A. Choudhary, A. Patra, B. V. Ashwinkumar, K. Srinathan, and C. Pandu Rangan. Perfectly Reliable and Secure Communication Tolerating Static and Mobile Mixed Adversary. In R. Safavi-Naini, editor, *Information Theoretic Security, Third International Conference, ICITS 2008, Calgary, Canada, August 10-13, 2008, Proceedings*, volume 5155 of *Lecture Notes in Computer Science*, pages 137–155. Springer, 2008.
- [20] A. Choudhary, A. Patra, Ashwinkumar B. V, K. Srinathan, and C. Pandu Rangan. On minimal connectivity requirement for secure message transmission in asynchronous networks. In V. Garg, R. Wattenhofer, and K. Kothapalli, editors, *Distributed Computing and Networking, 10th International Conference, ICDCN 2009, Hyderabad, India, January 03-06, 2009*, volume 5408 of *Lecture Notes in Computer Science*, pages 148–162, 2009.

- [21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2002.
- [22] T. H. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2004.
- [23] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multi-party computations secure against an adaptive adversary. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 311–326. Springer, 1999.
- [24] Y. Desmedt and Y. Wang. Perfectly secure message transmission revisited. In E. Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 502–517. Springer, 2003.
- [25] Y. Desmedt, Y. Wang, and M. Burmester. A complete characterization of tolerable adversary structures for secure point-to-point transmissions without feedback. In *Algorithms and Computation, 16th International Symposium, ISAAC 2005, Sanya, Hainan, China, December 19-21, 2005, Proceedings*, volume 3827 of *Lecture Notes in Computer Science*, pages 277–287. Springer Verlag, 2005.
- [26] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
- [27] D. Dolev. The Byzantine generals strike again. *Journal of Algorithms*, 3:14–30, 1982.
- [28] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *JACM*, 40(1):17–47, 1993.
- [29] M. J. Fischer, N. A. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *JACM*, 32(2):374–382, 1985.
- [30] M. Fitzi, M. K. Franklin, J. A. Garay, and S. H. Vardhan. Towards optimal and efficient perfectly secure message transmission. In S. P. Vadhan, editor, *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, volume 4392 of *Lecture Notes in Computer Science*, pages 311–322. Springer, 2007.
- [31] M. Fitzi, M. Hirt, and U. M. Maurer. Trading correctness for privacy in unconditional multi party computation. In H. Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, *Lecture Notes in Computer Science*. Springer, 1998.
- [32] Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Proactive RSA. In B. S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 440–454. Springer Verlag, 1997.

- [33] M. Franklin and R. Wright. Secure communication in minimal connectivity models. *Journal of Cryptology*, 13(1):9–30, 2000.
- [34] M. Franklin and M. Yung. Secure hypergraphs: Privacy from partial broadcast. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, pages 36–44, 1995.
- [35] M. K. Franklin and M. Yung. Secure Hypergraphs: Privacy from partial broadcast. *SIAM J. Discrete Math.*, 18(3):437–450, 2004.
- [36] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The Round complexity of verifiable secret sharing and secure multicast. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 580–589. ACM, 2001.
- [37] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, USA*, pages 218–229. ACM, 1987.
- [38] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public key and signature systems. In *CCS '97, Proceedings of the 4th ACM Conference on Computer and Communications Security, April 1-4, 1997, Zurich, Switzerland*, pages 100–110. ACM Press, 1997.
- [39] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In D. Coppersmith, editor, *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, volume 963 of *Lecture Notes in Computer Science*, pages 339–352. Springer Verlag, 1995.
- [40] W. C. Huffman and V. Pless. *Fundamentals of Error Correcting Codes*. Cambridge University Press, 2006.
- [41] M. V. N. A. Kumar, P. R. Goundan, K. Srinathan, and C. Pandu Rangan. On perfectly secure communication over arbitrary networks. In *PODC 2002, Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing, July 21-24, 2002 Monterey, California, USA*, pages 193–202. ACM, 2002.
- [42] K. Kurosawa and K. Suzuki. Truly efficient 2 round perfectly secure message transmission scheme. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 324–340. Springer, 2008.
- [43] K. Kurosawa and K. Suzuki. Almost secure (1-round,  $n$ -channel) message transmission scheme. *IEICE Transactions*, 92-A(1):105–112, 2009.
- [44] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
- [45] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error Correcting Codes*. North-Holland Publishing Company, 1978.
- [46] R. J. McEliece and D. V. Sarwate. On sharing secrets and reed solomon codes. *Communications of the ACM*, 24(9):583–584, 1981.

- [47] J. W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3(1):23–28, 1965.
- [48] T. K. Moon. *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley India, 2006.
- [49] A. Narayanan, K. Srinathan, and C. Pandu Rangan. Perfectly reliable message transmission. *Information Processing Letters*, 11(46):1–6, 2006.
- [50] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks (extended abstract). In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing, Montreal, Quebec, Canada, August 19-21*, pages 51–59. ACM Press, 1991.
- [51] C. Papadimitriou and U. Vazirani. *Discrete Mathematics for Computer Science*. Lecture Notes.
- [52] A. Patra, A. Choudhary, T. Rabin, and C. Pandu Rangan. The round complexity of verifiable secret sharing revisited. In S. Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 487–504. Springer Verlag, 2009.
- [53] A. Patra, A. Choudhary, and C. Pandu Rangan. Constant phase efficient protocols for secure message transmission in directed networks. In I. Gupta and R. Wattenhofer, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing, PODC 2007, Portland, Oregon, USA, August 12-15, 2007*, pages 322–323. ACM, 2007.
- [54] A. Patra, A. Choudhary, and C. Pandu Rangan. Unconditionally reliable and secure message transmission in directed networks revisited. In R. Ostrovsky, R. De Prisco, and I. Visconti, editors, *Security and Cryptography for Networks, 6th International Conference, SCN 2008, Amalfi, Italy, September 10-12, 2008. Proceedings*, volume 5229 of *Lecture Notes in Computer Science*, pages 309–326. Springer, 2008.
- [55] A. Patra, A. Choudhary, and C. Pandu Rangan. Brief announcement: Perfectly secure message transmission in directed networks revisited. In S. Tirthapura and L. Alvisi, editors, *Proceedings of the 28th Annual ACM Symposium on Principles of Distributed Computing, PODC 2009, Calgary, Alberta, Canada, August 10-12, 2009*, pages 305–328. ACM, 2009.
- [56] A. Patra, A. Choudhary, and C. Pandu Rangan. Round efficient unconditionally secure MPC and multiparty set intersection with optimal resilience. In B. K. Roy and N. Sendrier, editors, *Progress in Cryptology - INDOCRYPT 2009, 10th International Conference on Cryptology in India, New Delhi, India, December 13-16, 2009. Proceedings*, volume 5922 of *Lecture Notes in Computer Science*, pages 398–417. Springer Verlag, 2009.
- [57] A. Patra, A. Choudhary, and C. Pandu Rangan. On communication complexity of secure message transmission in directed networks. In K. Kant, S. V. Premmaraju, K. M. Sivalingam, and J. Wu, editors, *Distributed Computing and Networking, 11th International Conference, ICDCN 2010, Kolkata, India, January 3 - 6, 2010. Proceedings*, volume 5935 of *Lecture Notes in Computer Science*, pages 42–53. Springer Verlag, 2010.

- [58] A. Patra, A. Choudhary, K. Srinathan, and C. Pandu Rangan. Constant phase bit optimal protocols for perfectly reliable and secure message transmission. In R. Barua and T. Lange, editors, *Progress in Cryptology - INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11-13, 2006, Proceedings*, volume 4329 of *Lecture Notes in Computer Science*, pages 221–235. Springer, 2006.
- [59] A. Patra, A. Choudhary, K. Srinathan, and C. Pandu Rangan. Unconditionally reliable and secure message transmission in undirected synchronous networks: Possibility, feasibility and optimality. Accepted for publication in IJACT. A preliminary version appeared in [78], 2009.
- [60] A. Patra, A. Choudhary, M. Vaidyanathan, and C. Pandu Rangan. Efficient perfectly reliable and secure message transmission tolerating mobile adversary. In Y. Mu, W. Susilo, and J. Seberry, editors, *Information Security and Privacy, 13th Australasian Conference, ACISP 2008, Wollongong, Australia, July 7-9, 2008, Proceedings*, volume 5107 of *Lecture Notes in Computer Science*, pages 170–186. Springer, 2008.
- [61] A. Patra, A. Choudhury, and C. Pandu Rangan. Statistically reliable and secure message transmission in directed networks. Cryptology ePrint Archive, Report 2008/262. A preliminary version appeared as [54].
- [62] A. Patra, B. Shankar, A. Choudhary, K. Srinathan, and C. Pandu Rangan. Perfectly secure message transmission in directed networks tolerating threshold and non threshold adversary. In F. Bao, S. Ling, T. Okamoto, H. Wang, and C. Xing, editors, *Cryptology and Network Security, 6th International Conference, CANS 2007, Singapore, December 8-10, 2007, Proceedings*, volume 4856 of *Lecture Notes in Computer Science*, pages 80–101. Springer, 2007.
- [63] M. Pease, R. E. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *JACM*, 27(2):228–234, 1980.
- [64] B. Prabhu, K. Srinathan, and C. Pandu Rangan. Trading players for efficiency in unconditional multiparty computation. In S. Cimato, C. Galdi, and G. Persiano, editors, *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers*, volume 2576 of *Lecture Notes in Computer Science*, pages 342–353. Springer Verlag, 2002.
- [65] M. O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128–138, 1980.
- [66] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 73–85. ACM, 1989.
- [67] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.
- [68] R. M. Roth. *Introduction to Coding Theory*. Cambridge University Press, 2004.
- [69] H. Sayeed and H. Abu-Amara. Perfectly secure message transmission in asynchronous networks. In *Proceedings of 7th IEEE Symposium on Parallel and Distributed Processing*, pages 100–105. IEEE, 1995.

- [70] H. Sayeed and H. Abu-Amara. Efficient perfectly secure message transmission in synchronous networks. *Information and Computation*, 126(1):53–61, 1996.
- [71] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
- [72] B. Shanker, P. Gopal, K. Srinathan, and C. Pandu Rangan. Unconditional reliable message transmission in directed networks. In S. Teng, editor, *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 1048–1055. SIAM, 2008.
- [73] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(379):379–423 & 623–656, 1948.
- [74] P.W. Shor. Polynomial time algorithms for Prime factorization and Discrete Logarithms on a Quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
- [75] K. Srinathan. Secure distributed communication. PhD Thesis, IIT Madras, 2006.
- [76] K. Srinathan, M. V. N. Ashwin Kumar, and C. Pandu Rangan. Asynchronous secure communication tolerating mixed adversaries. In Y. Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 224–242. Springer, 2002.
- [77] K. Srinathan, A. Narayanan, and C. Pandu Rangan. Optimal perfectly secure message transmission. In M. K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 545–561. Springer, 2004.
- [78] K. Srinathan, A. Patra, A. Choudhary, and C. Pandu Rangan. Probabilistic perfectly reliable and secure message transmission - possibility, feasibility and optimality. In K. Srinathan, C. Pandu Rangan, and M. Yung, editors, *Progress in Cryptology - INDOCRYPT 2007, 8th International Conference on Cryptology in India, Chennai, India, December 9-13, 2007, Proceedings*, volume 4859 of *Lecture Notes in Computer Science*, pages 101–122. Springer, 2007.
- [79] K. Srinathan, A. Patra, A. Choudhary, and C. Pandu Rangan. Unconditionally reliable message transmission in directed hypergraphs. In M. K. Franklin, L. C. K. Hui, and D. S. Wong, editors, *Cryptology and Network Security, 7th International Conference, CANS 2008, Hong-Kong, China, December 2-4, 2008. Proceedings*, volume 5339 of *Lecture Notes in Computer Science*, pages 285–303. Springer, 2008.
- [80] K. Srinathan, A. Patra, A. Choudhary, and C. Pandu Rangan. Unconditionally secure message transmission in arbitrary directed synchronous networks tolerating generalized mixed adversary. In W. Li, W. Susilo, U. K. Tupakula, R. Safavi-Naini, and V. Varadharajan, editors, *Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, Sydney, Australia, March 10-12, 2009*, pages 171–182. ACM, 2009.

- [81] K. Srinathan, N. R. Prasad, and C. Pandu Rangan. On the optimal communication complexity of multiphase protocols for perfect communication. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pages 311–320. IEEE Computer Society, 2007.
- [82] K. Srinathan, P. Raghavendra, and C. Pandu Rangan. On proactive perfectly secure message transmission. In J. Pieprzyk, H. Ghodosi, and E. Dawson, editors, *Information Security and Privacy, 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007, Proceedings*, volume 4586 of *Lecture Notes in Computer Science*, pages 461–473. Springer, 2007.
- [83] K. Srinathan and C. Pandu Rangan. Efficient asynchronous secure multiparty distributed computation. In B. K. Roy and E. Okamoto, editors, *Progress in Cryptology - INDOCRYPT 2000, First International Conference in Cryptology in India, Calcutta, India, December 10-13, 2000, Proceedings*, volume 1977 of *Lecture Notes in Computer Science*, pages 117–129. Springer Verlag, 2000.
- [84] K. Srinathan and C. Pandu Rangan. Possibility and complexity of probabilistic reliable communication in directed networks. In E. Ruppert and D. Malkhi, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*, pages 265–274. ACM Press, 2006.
- [85] K. Srinathan, V. Vinod, and C. Pandu Rangan. Brief announcement: efficient perfectly secure communication over synchronous networks. In *PODC 2003, Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing (PODC 2003), July 13-16, 2003, Boston, Massachusetts, USA*, page 252. ACM Press, 2003.
- [86] Y. Wang and Y. Desmedt. Secure Communication in Multicast Channels: The Answer to Franklin and Wright’s Question. *J. Cryptology*, 14(2):121–135, 2001.
- [87] Y. Wang and Y. Desmedt. Perfectly secure message transmission revisited. *IEEE Transactions on Information Theory*, 54(6), 2008.
- [88] Q. Yang and Y. Desmedt. Cryptanalysis of secure message transmission protocols with feedback. To appear in Proc. of ICITS. Also available as Cryptology ePrint Archive: Report 2009/632, 2009.
- [89] A. C. Yao. Protocols for secure computations. In *Proceedings of 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, 3-5 November 1982*, pages 160–164. IEEE Computer Society, 1982.