General Secure Multi-Party Computation from any Linear Secret-Sharing Scheme

Ronald Cramer* Ivan Damgård** Ueli Maurer***

Abstract. We show that verifiable secret sharing (VSS) and secure multi-party computation (MPC) among a set of n players can efficiently be based on *any* linear secret sharing scheme (LSSS) for the players, provided that the access structure of the LSSS allows MPC or VSS at all. Because an LSSS neither guarantees reconstructability when some shares are false, nor verifiability of a shared value, nor allows for the multiplication of shared values, an LSSS is an apparently much weaker primitive than VSS or MPC.
Our approach to secure MPC is generic and applies to both the information-theoretic and the cryptographic setting. The construction is based on 1) a formalization of the special multiplicative property of an LSSS that is needed to perform a multiplication on shared values, 2) an efficient generic construction to obtain from any LSSS a multiplicative LSSS for the same access structure, and 3) an efficient generic construction to build verifiability into every LSSS (always assuming that the adversary structure allows for MPC or VSS at all).
The protocols are efficient. In contrast to all previous information-theoretically secure protocols, the field size is not restricted (e.g, to be greater than n). Moreover, we exhibit adversary structures for which our protocols are polynomial in n while all previous approaches to MPC for non-threshold adversaries provably have super-polynomial complexity.

1 Introduction

This paper is a more complete verison of the paper by the same authors that appears in the proceedings of EuroCrypt 2000, Springer Verlag LNCS series.

Secure *multi-party computation* (MPC) can be defined as the problem of n players to compute an agreed function of their inputs in a secure way, where security means guaranteeing the correctness of the output as well as the privacy of the players' inputs, even when some players cheat. A key tool for secure MPC, interesting in its own right, is *verifiable secret sharing* (VSS): a dealer distributes a secret value s among the players, where the dealer and/or some of the players may be cheating. It is guaranteed that if the dealer is honest, then the cheaters obtain no information about s, and all honest players are later

* Aarhus University, BRICS (Basic Reseach in Computer Science, center of the Danish National Research Foundation), work done while employed at ETH Zurich. Email: `cramer@brics.dk`. Supported by the Swiss SNF, grant no. 5003-045293.
** Aarhus University, BRICS. Email: `ivan@daimi.aau.dk`
*** ETH Zurich. Email: `maurer@inf.ethz.ch`. Supported by the Swiss SNF.

able to reconstruct s. Even if the dealer cheats, a unique such value s will be determined and is reconstructible without the cheaters' help.

It is common to model cheating by considering an *adversary* who may corrupt some subset of the players. One can distinguish between passive and active corruption. *Passive* corruption means that the adversary obtains the complete information held by the corrupted players, but the players execute the protocol correctly. *Active* corruption means that the adversary takes full control of the corrupted players. It is (at least initially) unknown to the honest players which subset of players is corrupted. Trivially, secure MPC is impossible if any subset can be corrupted. The adversary's corruption capability is characterized by an *adversary structure* [26] $\mathcal{A}$, a family of subsets where the adversary can corrupt any subset in $\mathcal{A}$. This is called an $\mathcal{A}$-adversary. The adversary structure could for instance consist of all subsets with cardinality less than some threshold value t. Of course, an adversary structure must be monotone, i.e. if $A \in \mathcal{A}$ and $B \subset A$, then $B \in \mathcal{A}$.

Both passive and active adversaries may be *static*, meaning that the set of corrupted players is chosen once and for all before the protocol starts, or *adaptive* meaning that the adversary can at any time during the protocol choose to corrupt a new player based on all the information he has at the time, as long as the total corrupted set is in $\mathcal{A}$.

Two basic models of communication have been considered in the literature. In the *cryptographic* model [25], all players are assumed to have access to messages exchanged between players, and hence security can only be guaranteed in a cryptographic sense, i.e. assuming that the adversary cannot solve some computational problem. In the *information-theoretic* (abbreviated i.t., sometimes also called secure channels) model [6, 11], it is assumed that the players can communicate over pairwise secure channels, and security can then be guaranteed even when the adversary has unbounded computing power.

An MPC protocol simulates an ideal setting in which the players give their inputs to a trusted party who computes the result and gives it back to the players. Security means that whatever an adversary can do in the real protocol he could essentially also do in the ideal setting. This assures both privacy and correctness. There are several technically different proposals for formalizing this (see e.g. [2, 29, 9]). While either definition could be used for a formal security proof of the protocols in this paper, any such proof would by far exceed the space limitations. Instead, we include sketches of proofs, generic enough to fit any of the definitions.

The outline of the paper is as follows: In Section 2 we review come previous work. In Section 3 we introduce some terminology and concepts, state the results and explain the role they play in comparison with earlier results. The technical results on LSSS are proved in Section 4. The protocols we propose are described in sections 5, 6, 7 and 8. Finally some issues relating to the complexity of our protocols are treated in Section 9.

2 Previous Work

The classical MPC results in the information-theoretic model due to Ben-Or, Goldwasser and Wigderson [6] and Chaum, Crépeau and Damgård [11] who showed that every function can be securely computed in presence of an adaptive, passive adversary, resp. an adaptive, active adversary if and only if the adversary corrupts less than $n/2$, resp. less than $n/3$ players. When a broadcast channel is available, and one accepts a non-zero probability that the protocol computes incorrect results, then one can tolerate less than $n/2$ active cheaters [32, 31].

The most general previous results for the cryptographic model are by Goldreich, Micali and Wigderson [25] who showed that, assuming trapdoor one-way permutations exist, any function can be securely computed in presence of a static, active adversary corrupting less than $n/2$ players and by Canetti et al. who show [10] that security against adaptive adversaries in the cryptographic model can also be obtained. VSS was introduced in [12].

All results mentioned so far only apply to threshold adversary structures. Gennaro [23] considered VSS in a non-threshold setting, and Hirt and Maurer [26] introduced the concept of an adversary structure and characterized exactly for which adversary structures VSS and secure MPC is possible. Let Q^2, resp. Q^3 be the conditions on an adversary structure that *no two*, resp. *no three* of the sets in the structure cover the full player set $\mathcal{P}$. The result of [26] can then be stated as follows: In the information-theoretic scenario, every function can be securely computed in presence of an adaptive, passive $\mathcal{A}$-adversary, resp. an adaptive, active $\mathcal{A}$-adversary if and only if $\mathcal{A}$ is Q^2, resp. $\mathcal{A}$ is Q^3. Beaver and Wool [3] propose a somewhat more efficient protocol for the passive case. The threshold results of [6], [11], [25] and [31] are special cases, where the adversary structure contains all sets of size less than $n/2$ or $n/3$.

This general model leads to strictly stronger results. For instance, in the case of 6 players $\{P_1, \ldots, P_6\}$ and active corruption, one can obtain a protocol secure against the structure with maximal sets $\{\{P_1\}, \{P_2, P_4\}, \{P_2, P_5, P_6\}, \{P_3, P_5\}, \{P_3, P_6\}, \{P_4, P_5, P_6\}\}$, whereas threshold type results tolerate only active cheating by a single player.

3 Results of this Paper

In this paper, we consider linear secret sharing schemes (LSSS). An LSSS is defined over a finite field K, and the secret to be distributed is an element in K. Each player receives from the dealer a share consisting of one or more field elements, each share is computed as a fixed linear function of the secret and some random field elements chosen by the dealer [1]. The *size* of an LSSS is the total number of field elements distributed. Only certain subsets of players, the *qualified* sets, can reconstruct the secret from their shares. *Unqualified* sets have no information about the secret. The collection of qualified sets is called the

[1] A seemingly weaker definition requires only that the reconstruction process be linear, however, this is essentially equivalent to the definition given here [4].

access structure of the LSSS, and the collection of unqualified sets is called the *adversary structure*.

Most proposed secret sharing schemes are linear, but the concept of an LSSS was first considered in its full generality by Karchmer and Wigderson who introduced the equivalent notion of Monotone Span Programs (MSP) which we describe in detail later. MSP's and LSSS's are in natural 1-1 correspondence.

The main goal in our paper is to provide an efficient construction which from any LSSS with adversary structure $\mathcal{A}$ builds MPC and VSS protocols secure against $\mathcal{A}$-adversaries (whenever this is possible). There are several motivations for this. First, basing VSS and MPC on as simple and weak a primitive as possible can help us design simpler and more efficient protocols because it is easier to come up with an implementation of a simpler primitive. Indeed, a wide range of general techniques for designing secret sharing schemes are known, e.g., Shamir [33], Benaloh-Leichter [5], Ito et al. [27], Bertilsson and Ingemarsson [7], Brickell [8] and van Dijk [17]. All these techniques result in LSSS's, and therefore are directly applicable to VSS and MPC by our results. Secondly, since LSSS's can be designed for any adversary structure, our approach allows us to build protocols handling any adversary structure for which VSS and MPC is possible at all. For some adversary structures this provably leads to an exponentially large efficiency improvement over known techniques, as we shall see.

We first give a brief overview of our basic approach: consider first the case where the adversary is passive. It is then trivial to add secrets securely: Each player holding an input shares it using the given LSSS, and each player adds up the shares he holds. By linearity of the LSSS, this results in a set of shares of the desired result.

Therefore, to do general MPC, it will suffice to implement multiplication of shared secrets. That is, we need a protocol where each player initially holds shares of secrets a and b, and ends up holding a share of ab. Such protocols are described for the threshold case in [25, 6, 11] and more recently in [24], based on Shamir's secret sharing scheme. We show below that the latter generalizes to work for any LSSS, provided that the LSSS is what we call *multiplicative*.

Loosely speaking, an LSSS is multiplicative if each player P_i can, from his shares of secrets a and b, compute a value c_i, such that the product ab can be computed as a linear combination of all the c_i's. It is *strongly* multiplicative if ab can be obtained using only values from honest players (we give a precise definition later).

With these techniques, using a multiplicative LSSS to implement passively secure MPC is quite straightforward. However, the multiplication property seems to require a very special structure in the LSSS. Nevertheless we show, perhaps somewhat surprisingly, that multiplicativity can be assumed without loss of generality: we give an efficient procedure that transforms any LSSS into a multiplicative LSSS of size at most twice that of the original one.

Finally, we consider the case of an active adversary. Basically, the same techniques as for the passive case will apply, provided we can build a linear *verifiable* secret sharing scheme from any given LSSS. We show that this can be done given a commitment scheme with certain convenient homomorphic properties. And we then build such a commitment scheme based also on the LSSS. With this VSS and the techniques described earlier for multiplication, an MPC protocol for active adversaries follows easily.

Thus, for the i.t. scenario, our main results are as follows:

THEOREM 1 *For any field K and any LSSS $\mathcal{M}$ with Q^3 adversary structure $\mathcal{A}$, there exists an error-free VSS protocol in the information-theoretic scenario, secure against any active and adaptive $\mathcal{A}$-adversary. The protocol has complexity polynomial in the size of $\mathcal{M}$ and* $\log |K|$.

THEOREM 2 *For any field K, any arithmetic circuit C over K, and any LSSS $\mathcal{M}$ with Q^2 adversary structure $\mathcal{A}$, there is an error-free MPC protocol computing C in the information theoretic scenario, secure against any adaptive and passive $\mathcal{A}$-adversary. The complexity of the protocol is polynomial in $|C|$,* $\log |K|$, *and the size of $\mathcal{M}$.*

THEOREM 3 *For any field K, any arithmetic circuit C over K, and any LSSS $\mathcal{M}$ with Q^3 adversary structure $\mathcal{A}$, there is an MPC protocol computing C in the information-theoretic scenario, secure against any adaptive and active $\mathcal{A}$-adversary. The complexity of the protocol is polynomial in $|C|$,* $\log |K|$, *the size of $\mathcal{M}$ and a security parameter k, where the error probability is exponentially small in k. If $\mathcal{M}$ is strongly multiplicative, there exists an error-free protocol for the same purpose, with complexity polynomial in $|C|$,* $\log |K|$ *and the size of $\mathcal{M}$.*

The statement of these results shows what can be done starting from a given LSSS. In practice, it may be that an adversary structure $\mathcal{A}$ is given by the application, and one wants the most efficient VSS or MPC possible for that structure. Our results show that we can build such protocols starting from any LSSS with a Q^3 (or Q^2) adversary structure containing $\mathcal{A}$. Such an LSSS always exists, by the results from Section 4. This leads naturally to a complexity measure for adversary structures, namely the size of the smallest LSSS that will work in this construction. From this perspective, our results show that the complexity of doing VSS/MPC secure for adversary structure $\mathcal{A}$ is upper bounded by the LSSS complexity of $\mathcal{A}$, up to a reduction polynomial in the number of players.

To compare our results to those of [26, 3] in terms of efficiency, we note that simple inspection of the protocols show that ours are more efficient by at least an additive polynomial amount for any non-threshold adversary structure. Moreover, the improvement can be much larger in some cases: we can show that there exists a family $\{\mathcal{A}_n\}_{n=1,2...}$ of adversary structures (where $\mathcal{A}_n$ is a structure on n players) for which our results lead to protocols that are polynomial time in n whereas any construction based on [26] or [3] has super-polynomial complexity.

A precise statement and proof of this result follows in Section 9. As an illustration, we describe a natural example of a family of structures, for which no previous solutions is known to work efficiently but for which linear size LSSS's can be built easily.

Suppose our player set is divided into two groups X and Y of m players each ($n = 2m$) where the players are on friendly terms within each group but tend to distrust players in the other group. Hence, a coalition of active cheaters might consist of almost all players from X or from Y, whereas a mixed coalition with players from both groups is likely to be quite small. Concretely, suppose we assume that a group of active cheaters can consist of at most $9m/10$ players from only X or only Y, *or* it can consist of less than $m/5$ players coming from both X and Y. This defines a Q^3 adversary structure, and so multi-party computations are possible in this scenario. Nevertheless, no threshold solution exists, since the largest coalitions of corrupt players have size more than $n/3$. It can be shown that no weighted threshold solution exists either for this scenario.

Note that it is trivial to give linear size monotone formulæ characterizing these structures (when arbitrary threshold functions are allowed as operators), and hence efficient LSSS's for these structures follow immediately by results from [5]. Therefore, our techniques can be used to build efficient MPC in these scenarios. No efficient construction is known using the protocols from [26, 3].

It is natural to ask if the results can be improved, i.e., can we base VSS/MPC on a even weaker primitive, for example an arbitrary secret sharing (SS) scheme? This would be the best we could hope for since VSS trivially implies SS. Recently, Cramer, Damgård and Dziembowski [14] have shown that while VSS can indeed be based on arbitrary SS schemes (by an efficient black-box reduction), there exists no black-box reduction reducing MPC to SS that is efficient on all relevant adversary structures. Thus, any generally efficient reduction would have to rely on special properties of the SS scheme, such as linearity. Hence, improving our MPC results in this direction seems like a highly non-trivial problem.

Remarkably, the situation for the cryptographic scenario is quite different. We have the following generalization of the threshold result from [25] (where the complexity of an SS scheme is defined as the complexity of distributing and reconstructing a secret):

Theorem 4 *Let C be an arithmetic circuit over a finite field K, let $\mathcal{A}$ be a Q^2 adversary structure, and let $\mathcal{S}$ be an SS scheme over K for which all sets in $\mathcal{A}$ are non-qualified and all complements of sets in $\mathcal{A}$ are qualified. If trapdoor one-way permutations exist, then there exists a secure MPC protocol computing C in the cryptographic scenario, secure against any active and static $\mathcal{A}$-adversary. It has complexity polynomial in $|C|$, the complexity of $\mathcal{S}$, and the security parameter k.*

The assumptions in this result are essentially minimal, but it does not lead to very practical protocols. However, if $\mathcal{S}$ is an LSSS and one-way group homomorphisms with specific extra properties exist, so-called q-one-way group homomorphisms [13], then very efficient protocols can be built. Particular assumptions sufficient for the existence of q-one way group homomorphisms include the RSA assumption, hardness of discrete logarithm in a group of prime order, or the decisional Diffie-Hellman assumption. As an example of what one obtains for the most efficient implementation of the primitives, we state the following:

Theorem 5 *Let C be an arithmetic circuit over $K = GF(q)$ for a k-bit prime q, and let $\mathcal{A}$ be a Q^2 adversary structure. If Diffie-Hellmann based probabilistic encryption in a group of order q is semantically secure, then there exists an MPC protocol computing C for the cryptographic scenario secure against any active and static $\mathcal{A}$-adversary. It has communication complexity $O(k \cdot |C|(\mu_{GF(q)}(\mathcal{A}))^2)$.*

Both the above results hold only for static adversaries. Security against adaptive adversaries can be obtained (at a loss of efficiency) by using non-committing encryption [10]. More details on this can be found in Section 8.

4 Multiplicative Monotone Span Programs

As mentioned earlier, Monotone Span Programs (MSP) are essentially equivalent to LSSS's (see e.g. [4]). It turns out to be convenient to describe our protocols in terms of MSP's, which we do for the rest of the paper. This section contains some basic definitions, notation and results relating to MSP's.

We first fix some notation for later use: The set of players in our protocols will be denoted by $\mathcal{P} = \{P_1, ..., P_n\}$. Consider any monotone Boolean function $f : \{0,1\}^n \to \{0,1\}$. By identifying subsets of $\mathcal{P}$ with their characteristic vectors[2], we can also apply f to subsets of $\mathcal{P}$. A set $S \subset \mathcal{P}$ for which $f(S) = 0$ (or $f(S) = 1$) is said to be *rejected* (or *accepted*) by f. The function f hence defines naturally an adversary structure, denoted $\mathcal{A}_f$, consisting of the sets rejected by f. Conversely, an adversary structure $\mathcal{A}$ defines a monotone function $f_\mathcal{A}$ rejecting exactly the sets in $\mathcal{A}$.

For two vectors $\mathbf{x}$ and $\mathbf{y}$ over a field K, $\mathbf{x} \otimes \mathbf{y}$ denotes the matrix whose i-th column is $x_i\mathbf{y}$, where x_i is the i-th coordinate of $\mathbf{x}$. If $\mathbf{x}$ and $\mathbf{y}$ have the same length, then $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the standard scalar product. A $d \times e$ matrix M defines a linear map from K^e to K^d. Im M denotes the image of this map, i.e. the subspace of K^d spanned by the columns of M. Ker M denotes the kernel of M, i.e. Ker $M = \{\mathbf{x} \in K^e : \; M\mathbf{x} = \mathbf{0}\}$. For the subspace V of a finite-dimensional vector space over K, the dual $V^\perp$ is defined as the subspace of vectors whose scalar product is 0 with all vectors in V. It is a basic fact from linear algebra that for any field K, $(V^\perp)^\perp = V$ and this immediately implies that $(\text{Ker } M)^\perp = \text{Im } (M^T)$, which we refer to as *duality argument.*

Definition 1 *A Monotone Span Program $\mathcal{M}$ is a triple (K, M, ψ), where K is a finite field, M is a matrix (with d rows and $e \leq d$ columns) over K and $\psi : \{1, \ldots, d\} \longrightarrow \{1, \ldots, n\}$ is a surjective function. The* size *of $\mathcal{M}$ is the number of rows (d).*

ψ labels each row with a number from $[1, \ldots, n]$ corresponding to a player in $\mathcal{P}$, so we can think of each player as being the "owner" of one or more rows.

[2] The characteristic vector of a set S is a vector in $\{0,1\}^n$ whose i-th component is 1 if and only if $P_i \in S$.

In the following, if M is the matrix of an MSP, and A is a subset of the players, then M_A denotes M restricted to those rows i with $\psi(i) \in A$. Similarly, if $\mathbf{x}$ is a d-vector, then $\mathbf{x}_A$ denotes $\mathbf{x}$ restricted to the coordinates i with $\psi(i) \in A$.

$\mathcal{M}$ yields a linear secret sharing scheme as follows: to distribute $s \in K$ the dealer chooses a random vector $\boldsymbol{\rho} \in K^{e-1}$ and writes $\mathbf{b} := (s, \boldsymbol{\rho})$. For each row $\mathbf{x}$ in M, the scalar product $\langle \mathbf{x}, \mathbf{b} \rangle$ is given to the owner of $\mathbf{x}$. We will denote the d-vector thus distributed by $M(s, \boldsymbol{\rho})$. It turns out that a set of players can reconstruct s precisely if the rows they own contain in their linear span the *target vector* of $\mathcal{M}$ which we have here globally fixed to be $(1, 0, ..., 0)$ (without loss of generality). Otherwise they get no information on s (see [28] for a proof of this). We note that the size of $\mathcal{M}$ is also the size of the corresponding LSSS.

The function computed by $\mathcal{M}$ is the monotone function accepting precisely those subsets that can reconstruct the secret [28]. It is well-known that for every field K, every monotone Boolean function is computed by some MSP over K. For a monotone function f, $\mathrm{msp}_K(f)$ will denote the size of the smallest MSP over K computing f. We refer to [20] for a characterization of MSP complexity in terms of certain combinatorial structures.

We now look at doing multiplication of values shared using MSP's. If secrets a and b have been shared using Shamir's secret sharing scheme to obtain shares $(a_1, .., a_n)$ and $(b_1, ..., b_n)$, respectively, it is immediate (see [24]) that ab can be computed as a linear combination of the values $a_i b_i$, where each such value can be computed locally by a single player. This can be generalized to LSSS's based on MSP's:

Given two d-vectors $\mathbf{x} = (x_1, ...x_d), \mathbf{y} = (y_1, ..., y_d)$, we let $\mathbf{x} \diamond \mathbf{y}$ be the vector containing all entries of form $x_i \cdot y_j$, where $\psi(i) = \psi(j)$. Thus, if d_i is the number of rows owned by player i, then $\mathbf{x} \diamond \mathbf{y}$ has $m = \sum_i d_i^2$ entries. Hence if $\mathbf{x}$ and $\mathbf{y}$ are the share-vectors resulting from sharing two secrets using $\mathcal{M}$, then each component of the vector $\mathbf{x} \diamond \mathbf{y}$ can be computed locally by some player.

Definition 2 *A* multiplicative *MSP is an MSP $\mathcal{M}$ for which there exists an m-vector $\mathbf{r}$, called a* recombination vector*, such that for any two secrets s, s' and any $\boldsymbol{\rho}$, $\boldsymbol{\rho'}$, it holds that*

$$s \cdot s' = \langle \mathbf{r}, M(s, \boldsymbol{\rho}) \diamond M(s', \boldsymbol{\rho'}) \rangle.$$

We say that $\mathcal{M}$ is strongly multiplicative *if for any player subset A that is rejected by $\mathcal{M}$, $\mathcal{M}_{\overline{A}}$ is multiplicative.*

The case of strong multiplication generalizes the threshold case with at most t corrupted players where we share secrets using polynomials of degree $t < n/3$. After multiplying points on two polynomials, the honest players can reconstruct the product polynomial on their own.

We define $\mu_K(f)$ to be the size of the smallest multiplicative MSP over K with computing f (∞ if f cannot be computed). Similarly, $\mu_K^*(f)$ is the complexity of f using strongly multiplicative MSP's. By definition, we have $\mathrm{msp}_K(f) \leq \mu_K(f) \leq \mu_K^*(f)$. We now characterize the functions that (strongly) multiplicative

MSP's can compute, and show that the multiplication property for an MSP can be assumed without loss of efficiency.

THEOREM 6 *For every finite field K and every monotone function f we have $\mu_K(f) < \infty$ if and only if f is Q^2, and $\mu^*_K(f) < \infty$ if and only if f is Q^3.*

THEOREM 7 *There exists an efficient algorithm which, on input an MSP $\mathcal{M}$ computing a Q^2 function f, outputs a multiplicative MSP $\mathcal{M}'$ (over the same field) computing f and of size at most twice that of $\mathcal{M}$. In particular $\mu_K(f) \leq 2 \cdot msp_K(f)$ for any K and f.*

We do not know if a similar result is true for strongly multiplicative MSP's. But Lemma 2 below gives a upper bound on their size, and give methods for constructing strongly multiplicative MSP's.

PROOF OF THEOREM 7. We make some observations first. Let f_0 and f_1 be monotone functions, computed by MSP's $\mathcal{M}_0 = (K, M_0, \psi)$ and $\mathcal{M}_1 = (K, M_1, \psi)$, respectively, where M_0 and M_1 are $d \times e$ matrices, where the mapping ψ is identical for both MSPs, and where the target vector is $\boldsymbol{t} = (1, 0, \ldots, 0)$. Now suppose that the matrices M_0 and M_1 satisfy

$$M_0^T M_1 = E, \tag{1}$$

where E is $e_o \times e_1$ matrix that is zero everywhere, except in its upper-left corner where the entry is 1. Next we prove the following claim.

CLAIM: From MSP's $\mathcal{M}_0$ and $\mathcal{M}_1$ as defined above, one can construct a multiplicative MSP computing $f_0 \vee f_1$ of size at most $2d$.

PROOF OF CLAIM: Consider the following straightforward LSSS. The dealer shares the secret $s \in K$ using LSSS_0 *and* LSSS_1, given by $\mathcal{M}_0$ and $\mathcal{M}_1$, respectively. That is, he selects a pair of vectors $(\mathbf{b}_0, \mathbf{b}_1)$ at random, except that the first entries are both s: $\langle \boldsymbol{t}, \mathbf{b}_0 \rangle = \langle \boldsymbol{t}, \mathbf{b}_1 \rangle = s$. Then he computes the pair of vectors $(\mathbf{s}_0, \mathbf{s}_1) = (M_0\mathbf{b}_0, M_1\mathbf{b}_1)$, and sends for $i = 1, \ldots, n$ the i-th coordinates of $\mathbf{s}_0$ and $\mathbf{s}_1$ to player $P_{\psi(i)}$. It is clear that a subset A of the players can reconstruct s from their joint shares if and only if A is accepted by the function $f_0 \vee f_1$, i.e. A must be qualified w.r.t. either LSSS_0 or LSSS_1.

Now we look at multiplication. Assume that $s' \in K$ is a secret with full set of shares $(\mathbf{s}'_0, \mathbf{s}'_1) = (M_0\mathbf{b}'_0, M_1\mathbf{b}'_1)$, where $\langle \boldsymbol{t}, \mathbf{b}'_0 \rangle = \langle \boldsymbol{t}, \mathbf{b}'_1 \rangle = s'$. Let $\mathbf{s}_0 * \mathbf{s}'_1$ be the d-vector obtained by coordinate-wise multiplication of $\mathbf{s}_0$ and $\mathbf{s}'_1$. Then from (1) we have

$$\langle \mathbf{1}, \mathbf{s}_0 * \mathbf{s}'_1 \rangle = \mathbf{s}_0^T \mathbf{s}'_1 = \mathbf{b}_0^T M_0^T M_1 \mathbf{b}'_1 = \mathbf{b}_0^T E \mathbf{b}'_1 = ss', \tag{2}$$

where $\mathbf{1}$ denotes the all-one vector of appropriate length. Note that for each i, the shares in the i-th coordinate of $\mathbf{s}_0$ and the i-th coordinate of $\mathbf{s}'_1$ are held by the same player.

We now build an MSP $\mathcal{M}$ with a $2d$ by $2e$ matrix M as follows: first make a matrix M' filled in with M_0 in the upper left corner and M_1 in the lower right corner. Let $\mathbf{k}$ be the column in M' that passes through the first column of M_1. Add $\mathbf{k}$ to the first column of M' and delete $\mathbf{k}$ from the matrix. Let M be the result of this. The labeling of M is carried over in the natural way from $\mathcal{M}_0$ and $\mathcal{M}_1$. Clearly, $\mathcal{M}$ corresponds exactly to the LSSS we just constructed. It is clear that the vector $(\mathbf{s}_0, \mathbf{s}_1) \diamond (\mathbf{s}'_0, \mathbf{s}'_1)$ contains among its entries the entries of $\mathbf{s}_0 * \mathbf{s}'_1$. Thus the vector with 1's corresponding to these entries and 0's elsewhere can be used as recombination vector, which shows that $\mathcal{M}$ is multiplicative. This concludes the proof of the claim.

We are now ready to prove Theorem 7. Recall that the dual function f^* of f is defined by: $f^*(x) = \overline{f(\overline{x})}$. We assume in the following that f is Q^2, i.e. $f(x) = 0$ implies $f(\overline{x}) = 1$ and thus $f^*(x) = 0$. It follows that $f = f \vee f^*$.

Let $\mathcal{M} = (K, M, \psi)$ be an MSP computing f, with target vector equal to $\boldsymbol{t} = (1, 0, \ldots, 0)$. To build a multiplicative MSP for f, we apply the above claim. We set $f_0 = f$, $f_1 = f^*$ and $\mathcal{M}_0 = \mathcal{M}$. It is then sufficient to find $\mathcal{M}_1$ computing $f_1 = f^*$ so that the pair M_0, M_1 satisfies equation (1).

In [21] a construction is presented which, given an MSP $\mathcal{N} = (K, N, \psi)$ of size d computing f (and with target vector $(1, \ldots, 1)$), yields a "dual" MSP $\mathcal{N}^* = (K, N^*, \psi)$ computing f^* (also with target vector $(1, \ldots, 1)$). The construction is as follows. N^* has also d rows and the same labeling as N and consists of one column for each set A accepted by f, namely a (reconstruction) vector $\boldsymbol{\lambda}$ satisfying $\boldsymbol{\lambda}^T N = (1, \ldots, 1)$ and $\boldsymbol{\lambda}_{\overline{A}} = \mathbf{0}$. The matrix N^* has generally exponentially many columns, but it is easy to see that any linearly independent generating subset of them (at most d) will also constitute a matrix of an MSP for the same access structure. It follows from the construction that $N^T N^*$ is an all-one matrix, which we call U.

In our case the target vector of $\mathcal{M}$ is $\boldsymbol{t} = (1, 0, \ldots, 0)$ instead of $(1, \ldots, 1)$, but the target vector can be transformed by adding the first column of M to every other column of M. More formally, let H be the isomorphism that sends an e-(column) vector to an e-(column) vector by adding its first coordinate to each other coordinate. Write $N = MH^T$. Then the MSP $\mathcal{N} = (K, N, \psi)$ is as $\mathcal{M}$ except that the target vector is all-one. Now let $\mathcal{N}^* = (K, N^*, \psi)$ be its dual MSP as constructed above. Finally write $M^* = N^*(H^{-1})^T$. Then $\mathcal{M}^* = (K, M^*, \psi)$ has target vector $\boldsymbol{t}$ and computes f^*. Observe that $M^T M^* = H^{-1} U (H^{-1})^T = E$, as desired. This proves that a suitable matrix M^* exists, but does not always give an efficient method for constructing it. However, the following method due to Serge Fehr [18] shows how construct M^* directly and efficiently from M: compute $\mathbf{w}_0$ be such that $M^T \mathbf{v}_0 = (1, 0, ..., 0)$, and compute $w_1, ..., w_{d-e}$ as a basis of the subspace $ker(M^T)$ (we may assume without loss of generality that $e \leq d$ and that the columns of M are linearly independent). Then let M^* be the matrix with columns $w_0, w_1, ..., w_{d-e}$. This is a matrix with d rows and $d - e + 1$ columns. Using the same labelling as for M we can turn this into an MSP, and it is elementary to show that this MSP computes f^* and satisfies $M^T M^* = E$. Theorem 7 follows. $\triangle$

Proof of Theorem 6. Since MSP's compute all monotone functions, it follows directly from this fact and Theorem 7 that every Q^2-function is computed by a multiplicative MSP. This also follows from secret sharing scheme used in [3], and this argument can be extended to prove that every Q^3-function is computed by a strongly multiplicative MSP. We conclude the proof by showing that multiplicative MSP's compute Q^2-functions. The proof in the Q^3-case is similar, and is omitted.

Let $\mathcal{M} = (K, M, \psi)$ be an MSP with target vector $\boldsymbol{t}$ computing a monotone boolean function f on n input bits, and let $\mathcal{A}_f$ be the adversary structure associated with f. Suppose that $\mathcal{M}$ is multiplicative, but that $\mathcal{A}_f$ is *not* Q^2. Thus, there exists a set $A \subset \{1, \ldots, n\}$ such that $A \cup \overline{A} = \{1, \ldots, n\}$ and $A, \overline{A} \in \mathcal{A}_f$. The latter implies that neither the rows of M_A nor those of $M_{\overline{A}}$ span $\boldsymbol{t}$. Hence, by the duality argument there exist vectors $\boldsymbol{\kappa}$ and $\boldsymbol{\kappa}'$, both with first coordinate equal to 1, such that $M_A\boldsymbol{\kappa} = \mathbf{0}$ and $M_{\overline{A}}\boldsymbol{\kappa}' = \mathbf{0}$. By the multiplication property, on one hand it follows that $\langle \mathbf{r}, M\boldsymbol{\kappa} \diamond M\boldsymbol{\kappa}' \rangle = 1$, where $\mathbf{r}$ is the recombination vector. But on the other hand, $M\boldsymbol{\kappa} \diamond M\boldsymbol{\kappa}' = \mathbf{0}$, by the choice of $\boldsymbol{\kappa}$, $\boldsymbol{\kappa}'$, and the fact that $A \cup \overline{A} = \{1, \ldots, n\}$, so this scalar product must be equal to 0: a contradiction. △

4.1 Constructing MSP's from adversary structures

Our main results show how to build VSS and MPC from a given MSP. The resulting protocols will be seure for whatever adversary structure the MSP happens to implement. However, in a practical situation, it will often be the case that an adversary structure is given by the application, and it is therefore of interest how one can build an MSP that is secure for a given adversary structure.

To this end, we will assume that the structure is described in terms of a monotone Boolean formula. Such formulas are usually defined to be Boolean formulas where only AND- and OR-operators occur. We will consider here a slightly different concept, namely we will allow arbitrary operators of form T^t_m, where T^t_m is the threshold function that outputs 1 if and only if at least t of its m inputs are 1 (AND and OR are special cases - we do not assume that m, t are the same for all operations in the formula) . Furthermore we will assume that no constants are used. We call this *generalized monotone formulae with no constants*, GMNC formulae for short.

Any such formula ϕ computes a monotone function f_ϕ on its input variables, this in turn defines an adversary structure $\mathcal{A}_\phi$ in the same way as we have seen earlier. It is clear that for any adversary structure, there exists a GMNC formula characterizing it in this way. If nothing else, it can always be constructed trivially from a complete list of the sets in the structure. But often, much smaller such formulae exist. In the following, $|\phi|$ denotes the size of a formula ϕ.

In [5], a construction is given which from a monotone formula constructs an LSSS secure for the corresponding adversary structure. This construction generalizes trivially to GMNC formulae, provided the field used is larger than the largest fan-in occurring in the formula. Basically this construction uses, for each operator T^t_m, an instance of Shamir's secret sharing scheme with m players and threhold t.

The construction results in an LSSS of size equal to the number of times the formula reads an input variable. An MSP can be obtained immediately from this. If one wants to work in a field K which not large enough, one can first do the construction over an extention field of K, and then use a technique appearing in [28] for constructing a new MSP over K. This blows up the size by at most a logarithmic factor, and so we have

Lemma 1 *There exists an efficient procedure which from any GMNC formula ϕ constructs an MSP (over any field K) which computes the function f_ϕ and has size $O(|\phi| \log |\phi|)$.*

In case f_ϕ is Q^2, we can invoke Theorem 7 to get a multiplicative MSP.

Unfortunately, the situation is not so nice for strongly multiplicative MSP's. We can, however, prove a corresponding result in this case if we place further restrictions on the formula.

We call a monotone formula *1/3-accepting* if each of the operators T^t_m satisfies $t-1 < m/3$. The complexity of a monotone function f in this model is denoted $\phi_{1/3}(f)$ (=∞ if it is not computable).

As an example, consider the formula on six variables:

$$\psi = T^2_4(x_1, x_3, T^2_4(x_1, x_2, x_3, x_4), T^2_4(x_1, x_2, x_5, x_6))$$

Then $\mathcal{A}_\psi$ is the example structure on 6 players mentioned in the introduction. We then have:

Lemma 2 *For any monotone Boolean function f we have: f is Q^3 iff $\phi_{1/3}(f) < \infty$. Moreover, from any $1/3$-accepting formula ϕ, one can efficiently construct a strongly multiplicative MSP $\mathcal{M}$ over field K computing f_ϕ, such that $\mathcal{M}$ has size at most $|\phi|$ if $|K| > |\phi|$, and otherwise the size of $\mathcal{M}$ is $O(|\phi| \log |\phi|)$.*

Proof. We first observe that if we use the above construction to build an MSP $\mathcal{M}$ from 1/3-accepting formula ϕ, then this MSP will automatically be strongly multiplicative. This follows from that fact that basic unit used in the construction is Shamir's scheme with parameters (t, m) such that $t-1 < m/3$, and such a scheme is already strongly multiplicative. It can then be shown by induction that MSP resulting from the Benaloh-Leichter construction is also strongly multiplicative.

From the above and Theorem 6 it now follows that functions computed by 1/3-accepting formulas are Q^3.

Finally, it can be shown by induction that each Q^3 function f is computable by a 1/3-accepting formula that consists only of OR- and 2-out-of-4-gates. This construction is exponential in general, but we can now conclude that the 1/3-accepting formulas compute exactly all of the Q^3-monotone functions.

5 Homomorphic Commitments and VSS

5.1 Preliminaries

We introduce some conventions and notation to be used in the protocol descriptions throughout the rest of the paper. We assume throughout (without loss of generality, and in accordance with the previous literature) that the function to be computed by $\{P_1, \ldots, P_n\}$ is given as an *arithmetic circuit* C of size $|C|$ over some finite field K, consisting of addition and multiplication gates. Our protocols are described making use of a broadcast channel. But note that in the i.t. scenario with an active adversary, we do *not* assume that such a channel is given for free as part of the model, however it can efficiently be simulated using the protocol of [19] that is secure against any given Q^3 adversary structure.

Let $\mathcal{M}$ be an MSP computing a Q^2 (or Q^3) function f. We will assume for simplicity that ψ is $1-1$, i.e. *each player owns exactly one row in* $\mathcal{M}$. In this case, $(a_1, ..., a_n) \diamond (b_1, ..., b_n) = (a_1 b_1, ..., a_n b_n)$. The generalization to many rows per player is straightforward, but would lead to rather complicated notation.

5.2 Overview of Commitments and Related Protocols

To prove Theorem 1, it is sufficient to construct, for each MSP $\mathcal{M} = (K, M, \psi)$ computing a Q^3 function f, an efficient VSS that is secure against an active $\mathcal{A}_f$-adversary. We first discuss generic primitives sufficient to construct an efficient VSS protocol and conclude by providing concrete realizations of these primitives.

A *commitment scheme* (for a given adversary structure $\mathcal{A}$) consists of two protocols: the protocol COMMIT allows a player P_i (the dealer) to commit to a value a and the protocol OPEN allows him later to reveal the committed value. The total information stored by the players after the protocol COMMIT is called the *commitment* and will be denoted $[a]_i$. Both protocols may be interactive protocols among the players and result either in the players accepting the outcome, or disqualifying the dealer. A commitment scheme must hide the committed value in the presence of an $\mathcal{A}$-adversary, and it must bind the dealer to the committed value, i.e. there is at most one value that the dealer can get accepted during the OPEN protocol. Both these properties can hold unconditionally, or relative to a computational assumption, depending on the scenario.

The crucial property we need is that commitments are *homomorphic*, which means that from commitments $[a]_i$ and $[b]_i$ the players can compute *without interaction* a commitment $[a+b]_i$ by P_i to the sum of the values, and that for a constant m they can compute $[ma]_i$. Thus, any linear function on committed values can be computed non-interactively. Homomorphic commitments have been used before in the context of zero-knowledge (e.g. [13]) and are implicit in some MPC protocols (e.g. [11]). We need two auxiliary protocols:

- A *commitment transfer protocol* (CTP) allows player P_i to transfer a commitment to player P_j (who of course learns a in the process), i.e. to convert $[a]_i$ into $[a]_j$. It must be guaranteed that this protocol leaks no information to the adversary if P_i and P_j are honest throughout the protocol, but also that the new commitment contains the same value as the old, even if P_i and P_j are both corrupt. It is therefore less trivial than one might expect.
- A *commitment sharing protocol* (CSP) allows player P_i to convert a committed value $[a]_i$ into a set of commitments to shares of a: $[a_1]_1, ..., [a_n]_n$, where $(a_1, ..., a_n) = M(a, \boldsymbol{\rho})$ for a random vector $\boldsymbol{\rho}$ chosen by P_i. This must hold

even if P_i is corrupt, and must leak no information to the adversary if P_i is honest throughout the protocol.

The CSP protocol is easy to describe at a general level: starting from $[a]_i$, P_i chooses a random vector $(\rho_1, ..., \rho_{e-1})$ and commits to $\rho_1, ..., \rho_{e-1}$, resulting in $[\rho_1]_i, ..., [\rho_{e-1}]_i$. Let $(a_1, ..., a_n)$ be the shares resulting from sharing a using the ρ_i's as random choices. Each a_i is a linear function of the committed values, and hence the players can compute $[a_1]_i, ..., [a_n]_i$ non-interactively. Finally, P_i uses CTP to convert $[a_j]_j$ into $[a_j]_i$, for $j = 1, \ldots, n$.

Committing to a and then performing CSP is equivalent to verifiably secret sharing (VSS) of a: the commitments to shares prevent corrupted players from contributing false shares when the secret is reconstructed. It remains to give efficient realizations of commitments and the CTP.

5.3 Realization of Commitments

To have a player D commit to a one could have him secret share a using $\mathcal{M}$. However, D may be corrupt and so must be prevented from distributing inconsistent shares. In the special case of threshold secret sharing, this means ensuring that all uncorrupted players hold points on a polynomial of bounded degree. For this purpose, we propose a protocol that can be seen as a generalization of the BGW-protocol from [6] where a bivariate polynomial was used[3]:

1. To commit to $s \in K$, D chooses a symmetric $e \times e$ matrix R at random, except that R has s in the upper left corner.[4] Let $\mathbf{v}_i$ be the row in M assigned to P_i and let $\mathbf{v}_i^T$ be its transpose (a column vector). Then D sends to P_i the vector $\mathbf{u}_i = R \cdot \mathbf{v}_i^T$. The share s_i of s given to P_i is defined to be the first entry of $\mathbf{u}_i$. Hence the product $\langle \mathbf{v}_j, \mathbf{u}_i \rangle := s_{ij}$ can be thought of as a share of s_i given to P_j. Note that we have $\langle \mathbf{v}_j, \mathbf{u}_i \rangle = \langle \mathbf{v}_j R, \mathbf{v}_i^T \rangle = \langle \mathbf{v}_i, R\mathbf{v}_j^T \rangle = \langle \mathbf{v}_i, \mathbf{u}_j \rangle$.
2. P_i sends to each P_j the value $\langle \mathbf{v}_j, \mathbf{u}_i \rangle$, who compares this to $\langle \mathbf{v}_i, \mathbf{u}_j \rangle$ and broadcasts a message *complaint*(i, j) if the values are not equal.
3. In response to *complaint*(i, j), D must broadcast the correct value of s_{ij}.
4. If any player P_i finds that the information broadcast by D does not match what he received from D in step 1, he broadcasts an *accusation*, thus claiming that D is corrupt.
5. In response to an accusation by P_i, D must broadcast all information sent to P_i in step 1.
6. Every player checks whether the information broadcast by D in the previous step is consistent with everything he has received from D previsously. If

[3] Apart from the threshold case [6], our protocol is a VSS, i.e. efficient reconstruction without the help of the dealer is possible, if for each set B whose complement is in $\mathcal{A}$, the matrix M_B has full rank. In this case, players should store all information received by the dealer to reconstruct efficiently. In general, however, we cannot guarantee efficient reconstruction, so we only use it here as a commitment scheme.

[4] One can think of step 1 in this protocol (choosing R) as corresponding to choosing a *symmetric* bivariate polynomial in the VSS protocol of [6].

this is not the case he broadcasts an accusation against D. If at this point, the set of accusing players is in $\mathcal{A}$, the commit protocol is accepted by the honest players, and accusing players accept the share broadcast for them by D. Otherwise the commit protocol is rejected.

To open a commitment, D broadcasts s and the full set of shares $\{s_i\}$, and each player broadcasts a binary message ("agree" or "complain"). If the shares consistently determine s and only a set of players in $\mathcal{A}$ complained, then the opening is accepted.

We now explain why this commitment scheme works. First, assume D remains honest throughout the commit protocol. To show that the adversary obtains no information about s, note first that steps 2-6 of the commit protocol are designed such that the adversary learns nothing he was not already told in step 1. Now let A be any set in $\mathcal{A}$, and let $M_A R$ denote the information received by the players in A in the commit phase, finally let X be any symmetric matrix satisfying the equation $M_A X = M_A R$, and having some $\tilde{s} \in K$ in its upper-left corner. Since A is rejected by $\mathcal{M}$, it follows by the duality argument that there exists a vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_e) \in \mathrm{Ker} M_A$ with $\mu_1 = 1$. Consider the matrix $\boldsymbol{\mu} \otimes \boldsymbol{\mu}$. Note that this matrix is symmetric and that it has 1 in its upper-left corner. Then $X + (s - \tilde{s})\boldsymbol{\mu} \otimes \boldsymbol{\mu}$ satisfies the equation as well, has s in its upper left corner and is symmetric. Hence, for each possible $\tilde{s}$, the number of different solutions X with $\tilde{s}$ in the upper left corner is the same, and hence the adversary learns no information on s in step 1. Finally note that if D remains honest throughout, all honest players will agree with him, so the opening always succeeds.

Now assume that D is corrupt and that the commit protocol has been accepted. Consider the full set of players and subtract the set of accusers and the set of corrupted players. By the Q^3 property, we know that the remaining set of (honest) players is not in $\mathcal{A}$, i.e. it is large enough to uniquely determine a secret shared by $\mathcal{M}$ (this is called a qualified set). Assume wlog that these are the first t players. These players have information that is consistent with each other and also with the information broadcast by the dealer (since they did not accuse the dealer). It follows that *every* pair of honest players agree on the value s_{ij} they have in common. And furthermore, that if P_i is honest, all the s_{ij}'s known to him are consistent with $\mathbf{u}_i$. Define the symmetric $n \times n$ matrix S to be the matrix containing all the s_{ij}'s known to players $P_1, ..., P_t$ (this leaves entries s_{ij} with $i, j > t$ undefined). For $i \leq t$, the i'th column determines s_i uniquely, as a fixed linear combination of the first t entries (since the first t players form a qualified set). The coefficients in this linear combination depend only on $\mathcal{M}$ and so are the same for any column. It follows that the row of shares $(s_1, ..., s_n)$ is determined as a linear combination of the first t rows of S. Since each of these rows consistently determines a secret (namely s_i for the i'th row), it follows by linearity of MSP secret sharing that the row $(s_1, ..., s_n)$ consistently determines some secret s.

It remains to be shown that opening the commitment must either reveal s or be rejected. Assume the opening is accepted. Then consider the full player set and subtract the set of corrupt players and the set of players who complained about the opening. The remaining set cannot be in $\mathcal{A}$ by the Q^3 property and so is qualified. It consists of honest players that did not complain, i.e. the shares revealed for them are the same as those received in the commitment phase. Hence the revealed value must be s.

5.4 Realization of the CTP

The following protocol converts $[a]_i$ into $[a]_j$:

1. Given a commitment $[a]_i$ realized as above with P_i in the role of D, P_i sends privately the shares determining a to P_j. If this information is not consistent, then P_j broadcasts a complaint, and the protocol continued at step 4.
2. P_j commits to a (independently), resulting in $[a]_j$.
3. Using linearity of commitments, P_j opens the difference $[a]_i - [a]_j$ to reveal 0, using the information from step 1 as if he created $[a]_i$ himself. If this succeeds, the protocol ends. Otherwise do Step 4.
4. If we arrive at this step, it is clear that at least one of P_i and P_j is corrupt, so P_i must then open $[a]_i$ in public, and we either disqualify P_i (if he fails) or continue with a default commitment to a assigned to P_j.

6 MPC Secure Against Passive Adversaries

To prove Theorem 2, it is sufficient to construct for each MSP $\mathcal{M} = (K, M, \psi)$ computing a Q^2 function f, an efficient protocol that is secure against a passive $\mathcal{A}_f$-adversary. By Theorem 7 we can assume without loss of generality (or efficiency) that $\mathcal{M}$ is multiplicative.

The protocol, which is a generalization of a threshold protocol appearing in [24], starts by letting each player share each of his inputs using $\mathcal{M}$ and send a share to each player. The given arithmetic circuit over K is then processed gate by gate, maintaining as invariant that all inputs and intermediate results are secret-shared, i.e. each such value $a \in K$ is shared (using $\mathcal{M}$) by shares $a_1, ..., a_n$, where P_i holds a_i. Moreover, if a depends on an input from an honest player, this must be a *random* set of shares with the only constraint that it determines a. At the beginning, only the input values are classified as having been computed. Once an output value x has been computed, it can be reconstructed in the obvious way by broadcasting the shares $x_1, ..., x_n$. It is therefore sufficient to show how addition and multiplication gates are handled. Assume the input values to a gate are a and b, determined by shares $a_1, ..., a_n$ and $b_1, ..., b_n$, respectively.

Addition For $i = 1, \ldots, n$, P_i computes $a_i + b_i$. The shares $a_1 + b_1, \ldots, a_n + b_n$ determine $a + b$ as required by the invariant.

Multiplication For $i = 1, \ldots, n$, P_i computes $a_i \cdot b_i = \tilde{c}_i$.
Resharing step: P_i secret shares $\tilde{c}_i$, resulting in shares $c_{i1}, ..., c_{in}$, and sends c_{ij} to player P_j.

Recombination step: For $j = 1, \ldots, n$, player P_j computes $c_j = \sum_{i=1}^{n} r_i c_{ij}$, where $(r_1, \ldots, r_n)$ is a fixed recombination vector of $\mathcal{M}$. The shares $c_1, \ldots, c_n$ determine $c = ab$ as required by the invariant.

We do not have space to prove formally the security of this protocol here. However, to get a feeling for why it is secure, note first that the addition and multiplication step compute correct results simply by linearity of the secret sharing, and by the multiplication property. To argue that privacy is maintained, the crucial point is to show that the sharing of a result c of the multiplication step starting from a, b is random with the only restriction that it determines c (the corresponding result for addition is trivial).

It is easily seen that the set of shares determining c can be written as $(c_1, ..., c_n) = M(c, \boldsymbol{\rho})$, where in fact $\boldsymbol{\rho} = \sum_{i=1}^{n} r_i \boldsymbol{\rho}_i$ and where $\boldsymbol{\rho}_i$ was chosen by P_i. Let $B = \{P_i |\ r_i \neq 0\}$. We claim that $B \notin \mathcal{A}$. Indeed, let $\mathcal{M}$ be an MSP with multiplication, and let $\mathbf{r}$ be a recombination vector. Then $B = \{P_i |\ r_i \neq 0\} \notin \mathcal{A}$. Towards a contradiction, suppose $B \in \mathcal{A}$. By the duality argument, choose $\boldsymbol{\kappa}$ such that $M_B \boldsymbol{\kappa} = \mathbf{0}$ and the first coordinate κ_1 of $\boldsymbol{\kappa}$ is 1. Then by definition of the multiplication property we have that $1 = \kappa_1^2 = \langle \mathbf{r}, M\boldsymbol{\kappa} \diamond M\boldsymbol{\kappa} \rangle$. But on the other hand, since $M_B \boldsymbol{\kappa} \diamond M_B \boldsymbol{\kappa} = \mathbf{0}$ and $\mathbf{r}_{\overline{B}} = \mathbf{0}$, this must be equal to 0, a contradiction. This proves the claim. Therefore, the choice of at least one $\boldsymbol{\rho}_i$ where $r_i \neq 0$ remains unknown to the adversary and is made randomly and independently of anything else. This can be used when building a simulator for an adversary: when he corrupts a player, what we have to do is essentially to come up with a random share for this player of each shared value. Each such share must be consistent with what the adversary already knows. By the above, this can be handled independently for each shared value, and so can be easily done by solving a system of linear equations.

7 MPC Secure Against Active Adversaries

To prove Theorem 3, it is sufficient to construct for each MSP $\mathcal{M} = (K, M, \psi)$ computing a Q^3 function f, an efficient protocol that is secure against an active $\mathcal{A}_f$-adversary. Since Q^3-functions are in particular Q^2, we can assume by Theorem 7 without loss of generality (or efficiency) that $\mathcal{M}$ is multiplicative.

Like in some previous protocols, the basic approach is to ensure that all players are committed to the values they hold, and to have them prove that they are performing their local operations correctly. In what follows, we use a generic commitment scheme and auxiliary protocols as in Section 5.

7.1 The CMP Protocol

We need an additional primitive, namely a *Commitment Multiplication Protocol* (CMP). Such a protocol starts from commitments $[a]_i, [b]_i, [c]_i$ and allows P_i to convince the other players that $ab = c$. If P_i is corrupted, then the honest players should accept the proof only if $ab = c$ (in the cryptographic scenario, an negligible error probability is allowed). If P_i remains honest, it must leak no information to the adversary beyond the fact that $ab = c$. Moreover, in the event that $[c]_i$ is opened, the adversary must learn nothing about a, b beyond what is implied by c and the other information he holds. The following CMP protocol is a generalization of a protocol suggested in [15] and works for any homomorphic commitment scheme.

1. Inputs are commitments $[a]_i, [b]_i, [c]_i$ where P_i claims that $ab = c$. P_i chooses a random β and makes commitments $[\beta]_i, [\beta b]_i$.
2. The other players jointly generate a random challenge r using standard techniques.
3. P_i opens the commitments $r[a]_i + [\beta]_i$ to reveal a value r_1. P_i opens the commitment $r_1[b]_i - [\beta b]_i - r[c]_i$ to reveal 0.
4. If any of these opening fail, the proof is rejected, else it is accepted.

It is easy to show that if P_i is honest, then all values opened are random (or fixed to 0) and so reveal no extra information to the adversary. Furthermore, if after committing in step 2, P_i can answer correctly two different challenges, then $ab = c$. Thus the error probability is at most $1/|K|$, and the protocol can be iterated to reach any desired error probability.

CMP with zero error For commitments based on MSP's, as we have seen earlier, one can construct an error-free CMP protocol, provided the MSP we have is strongly multiplicative:

In the threshold case, when values a, b have been shared using two polynomials of degree t, multiplying corresponding shares of a, b yields a sharing of ab, *but in a new secret sharing scheme*: we now need $2t + 1$ players in stead of $t + 1$ to reconstruct the secret. Something similar can be shown to hold for MSP-based secret sharing:

Let $\mathcal{M} = (K, M, \Psi)$ be an MSP with strong multiplication. It can be shown that there exists (an efficiently constructible) MSP $\mathcal{M}' = (K, M', \Psi')$ such that the following holds.

- For all $a, \boldsymbol{\rho}_a$ and $b, \boldsymbol{\rho}_b$, there exists a $\boldsymbol{\rho}$ such that $M(a, \boldsymbol{\rho}_a) \diamond M(b, \boldsymbol{\rho}_b) = M'(ab, \boldsymbol{\rho})$ (*)
- If a set A is rejected by $\mathcal{M}$ then A is also rejected by $\mathcal{M}'$, but $\overline{A}$ is accepted by $\mathcal{M}'$.

We omit the technical and fairly straightforward proof of this claim.

In other words, if $\mathbf{s}$ and $\mathbf{s}'$ are full sets of shares for a and b, respectively, then $\mathbf{s} \diamond \mathbf{s}'$ is a full set of shares of ab according to $\mathcal{M}$. Moreover, the secret ab is determined uniquely even if we delete from $\mathbf{s} \diamond \mathbf{s}'$ the shares that belong to a set A rejected by $\mathcal{M}$.

The CMP protocol is now as follows:

1. Inputs are commitments $[a]_i, [b]_i, [c]_i$ where P_i claims that $ab = c$. First P_i performs CSP on commitments $[a]_i, [b]_i$ to get committed shares $[a_1]_1, ..., [a_n]_n$ and $[b_1]_1, ..., [b_n]_n$.

2. P_i computes the vector $\boldsymbol{\rho}$ as defined in equation (*) above (easy by solving a set of linear equations) and commits to each entry of $\boldsymbol{\rho}$. From these commitments and $[c]_i$, the players compute (by linear operations) commitments to the shares of c under M', $[c_1]_i, ..., [c_n]_i$ where of course P_i claims that $a_j b_j = c_j$, for $1 \leq j \leq n$.
3. For $j = 1, \ldots, n$, commitment $[c_j]_i$ is openened privately to P_j, i.e. the shares needed to open it are sent to P_j (instead of being broadcast). If the value revealed this way is not $a_j b_j$, P_j broadcasts a complaint and opens (his own) commitments $[a_j]_j, [b_j]_j$. In response, P_i must open $[c_j]_i$ and is disqualified if $a_j b_j \neq c_j$.

Very briefly, this works since the protocol guarantees that $c_1, .., c_n$ are a set of shares of c under M'. Even if P_i is corrupt, he must have $a_j b_j = c_j$ for each honest P_j or be disqualified. But since the set of honest players can determine uniquely a secret shared under M', it follows that $ab = c$.

7.2 The General MPC Protocol

The general MPC protocol starts by asking each player to VSS each of his input values as described above: he commits to the value and then performs CSP. A player failing to execute this correctly is disqualified and we take default values for his inputs.

We then work our way through the given arithmetic circuit, maintaining as invariant that all inputs and intermediate results computed so far are VSS'ed as described above, i.e. each such value a is shared (using $\mathcal{M}$) by committed shares $[a_1]_1, ..., [a_n]_n$ where *all* these shares are correct, also those held by corrupted players. Moreover, if a depends on an input from an honest player, this must be a *random* set of shares with the only constraint that it determines a. At the beginning, only the input values are classified as having been computed.

Once an output value x has been computed, it can be reconstructed in the obvious way by opening commitments to the shares $x_1, ..., x_n$. This will succeed, as the honest players will contribute enough correct shares, and a corrupted player can only choose between contributing a correct share, or be disqualified by trying to open an incorrect value. It is therefore sufficient to show how addition and multiplication gates are handled. Assume the input values to a gate are a and b, determined by committed shares $[a_1]_1, ..., [a_n]_n$ and $[b_1]_1, ..., [b_n]_n$.

Addition For $i = 1, \ldots, n$, P_i computes $a_i + b_i$ and the players (non-interactively) compute $[a_i + b_i]_i$. By linearity of the secret sharing and commitments, $[a_1 + b_1]_1, ..., [a_n + b_n]_n$ determine $a + b$ as required by the invariant.

Multiplication For $i = 1..n$, P_i computes $a_i \cdot b_i = \tilde{c}_i$, commits to it, and performs CMP on inputs $[a_i]_i, [b_i]_i, [\tilde{c}_i]_i$.
Resharing step: P_i performs CSP on $[\tilde{c}_i]_i$, resulting in the commitments $[c_{i1}]_1, ..., [c_{in}]_n$. We describe below how to recover if P_i fails to execute this phase correctly.

Recombination step: For $j = 1..n$, player P_j computes $c_j = \sum_{i=1}^{n} r_i c_{ij}$, where $(r_1, ..., r_n)$ is a fixed recombination vector. Also all players compute (non-interactively) $[c_j]_j = \sum_{i=1}^{n} r_i [c_{ij}]_j = [\sum_{i=1}^{n} r_i c_{ij}]_j$. By the multiplication property and linearity of $\mathcal{M}$, the commitments $[c_1]_1, ..., [c_n]_n$ determine $c = ab$ as required by the invariant.

It remains to be described what should be done if a player P_i fails to execute the multiplication and resharing step above. In general, the simplest way to handle such failures is to go back to the start of the computation, open the input values of the players that have just been disqualified, and restart the computation, simulating openly the disqualified players. This allows the adversary to slow down the protocol by a factor at most linear in n. This protocol, together with the VSS and main MPC protocols described previously, are the basis for proving Theorem 3.

The described approach for dealing with cheaters can be used only for secure function evaluation, but not for an ongoing secure computation. For the latter, one can introduce an additional level of sharings: each value a player is committed to in the above description is in addition verifiably secret shared, this can be accomplished using the CSP protocol. Then, if a player fails, the honest players can immediately open all his secret values and simulate the failed player openly from that point.

8 The Cryptographic Scenario

The cryptographic scenario can be handled in various ways. One way is to start from the full information theoretically secure protocol for an active adversary and replace every transmission of a message (over a secure channel) by transmission over an open network of a non-committing encryption[10] of the same message. This will, by the results shown in [10] lead to an adaptively secure protocol for the cryptograhic scenario. This protocol will only be secure against a Q^3-adversary. However doing the same transformation on the Q^2 secure protocol from [15] (which is derived from a threshold protocol using ideas from this paper) will give us Q^2-security in the cryptographic scenario.

Current state of the art for non-committing encryption menas that these transformations result in rather inefficient protocols. If we are willing to settle for static security, a more efficient translation can be done by noting that the general MPC protocol outlined in section 7.2 can be used with no change in the cryptographic scenario, provided we can implement homomorphic commitments and a CTP protocol for the cryptographic scenario (note that the CSP and CMP (with eror) protocols we described earlier do not depend on the scenario).

Starting from this, we sketch a proof of Theorem 5. As commitments in this scenario, we use the commitments from [13] based on q-one-way homomorphisms, which exists, e.g. if RSA is hard to invert or if the decisional Diffie-Hellman problem in some prime order group is hard. We then require that the field over which we compute is $GF(q)$. A simple example is if we have primes p, q, where

$q|p-1$ and g, h, y are elements in Z_p^* of order q chosen as public key by player P_i. Then $[a]_i$ is of form $(g^r, y^a h^r)$, i.e. a Diffie-Hellman (El Gamal) encryption of y^a under public key g, h. In [13], protocols are shown for proving efficiently in zero-knowledge that you know the contents of a commitment, and that two commitments contains the same value, even if they were done w.r.t. different public keys. It is trivial to derive a CTP from this: P_i privately reveals the contents and random bits for $[a]_i$ to P_j. If this is not correct, P_j complains, otherwise he makes $[a]_j$ and proves it contains the same value as $[a]_i$.

We note that for technical reasons, in order to be able to do a simulation-based proof of security of the overall MPC protocol using these commitments, each player must prove knowledge of his secret key intially, as well as prove that he knows the contents of each commitment he makes. These commitments, together with the VSS and main MPC protocols described earlier, form the basis for proving Theorem 5.

To show Theorem 4, we need to depart a bit from the general framework we have used until now. We sketch the construction of the protocol claimed, omitting many technical details: First, note that the constructions of [25] imply that (cryptographic) commitment and oblivious transfer between any pair of players can be done under our assumption on existence of one-way trapdoor permutations. Hence, a player can VSS a value a by sharing it according to $\mathcal{S}$, encrypting each share under the public key of the receiver, and proving in zero-knowledge that this was done correctly. Thus, in the Input Distribution Stage, we have each player VSS his inputs in this way. For the computation stage, we use the fact that oblivious transfer is sufficient to do MPC with an arbitrary number of cheater, where however it cannot be guaranteed that the protocol completes successfully. Also this follows essentially from [25], an explicit reduction can be found in the later [16], which shows how to do MPC based on commitments and oblivious transfer as blackboxes. This protocol makes no assumption on the adversary structure, but on the other hand stops completely if a cheater is detected. In our case, however, such an event is easily handled: the players go back to the VSS's from the first stage, reconstruct the inputs of the cheater, and redo the computation without him.

9 Complexity Issues

In this section, we look at the consequences our reults have for the complexity of doing MPC secure against adversary structures in general. In other words, we ask: what is the most efficient MPC protocol secure for a given adversary structure $\mathcal{A}$ that our results can yield? and how does this compare to earlier constructions of MPC protocols?

It is clear from the above that for an active (passive) adversary our most efficient protocol will be the one that is derived from the smallest MSP whoose adversary structure is $Q^3(Q^2)$ and contains $\mathcal{A}$. In other words, to protect against an $\mathcal{A}$-adversary, it is enough to protect against an even stronger adversary who can corrupt more sets, and this might be easier, if this stronger adversary has a simpler description in terms of a smaller MSP.

Motivated by this, we define

Definition 3 *For all adversary structures $\mathcal{A}$:*
$msp_K(\mathcal{A}; Q^2) = min\{msp_K(f) | f \text{ is } Q^2 \text{ and } f \text{ rejects } \mathcal{A}\}$
$msp_K(\mathcal{A}; Q^3) = min\{msp_K(f) | f \text{ is } Q^3 \text{ and } f \text{ rejects } \mathcal{A}\}$.

Clearly, $\mathrm{msp}_K(\mathcal{A}; Q^2)$ $(\mathrm{msp}_K(\mathcal{A}; Q^3))$ is finite iff $\mathcal{A}$ is Q^2 (Q^3). With this notation we can say that the best protocols we can build for adversary structure $\mathcal{A}$ have complexity polynomial in $\mathrm{msp}_K(\mathcal{A}; Q^2)$ $(\mathrm{msp}_K(\mathcal{A}; Q^3))$.

A similar description of protocol complexity can be given for the constructions of Hirt and Maurer [26]. We defined earlier 1/3-accepting formulae as monotone formulae with no constants and containing only threshold operators T_m^t where $t-1 < m/3$. We define *majority-accepting formulae* to be the same, except that $t-1 < m/2$. Similarly to the complexity measures for MSP's, we let $\Phi_{1/3}(f)$ ($\Phi_{1/2}(f)$) be the size of the smallest 1/3-accepting (majority-accepting) formula computing f - ∞ if f cannot be computed. Then we define

Definition 4 *For all adversary structures $\mathcal{A}$:*
$\Phi_{1/2}(\mathcal{A}) = min\{\Phi_{1/2}(f) | f \text{ rejects } \mathcal{A}\}$
$\Phi_{1/3}(\mathcal{A}) = min\{\Phi_{1/3}(f) | f \text{ rejects } \mathcal{A}\}$

The point of this is that it now follows from simple inspection of the construction from [26] that the most efficient protocols they can build for adversary structure $\mathcal{A}$ and an active (passive) adversary have complexity polynomial in $\Phi_{1/3}(\mathcal{A})$ $(\Phi_{1/2}(\mathcal{A}))$. From this, Lemma 2, and inspection of the protocols, it follows that for any adversary structure, we can always do at least as well as [26]. But in some cases, we can do much better, as can be seen from the following theorem:

Theorem 8 *There exists families $\{f_n\}$ and $\{g_n\}$ of monotone Boolean functions on n variables such that every f_n is maximal Q^2 and every g_n is maximal Q^3. Furthermore $msp_{GF(2)}(f_n)$ and $msp_{GF(2)}(g_n)$ are $O(n)$, but both families require superpolynomial size monotone circuits, in particular $\phi_{ma}(f_n)$ and $\phi_{1/3}(g_n)$ are $n^{\Omega(\log n)}$.*

The fact that g_n is maximal Q^3, means that g_n is the only Q^3 function rejecting the adversary structure $\mathcal{A}_{g_n}$, and similarly for f_n. Hence we get immediately

Corollary 1 *Notation as in Theorem 8. Then we have that $msp_{GF(2)}(\mathcal{A}_{f_n}; Q^2)$, $msp_{GF(2)}(\mathcal{A}_{g_n}; Q^3)$ are linear in n, but $\phi_{1/2}(\mathcal{A}_{f_n})$, $\phi_{1/3}(\mathcal{A}_{g_n})$ are superpolynomial.*

Thus our MPC protocols for these adversary structures (and in particular our VSS construction for $\mathcal{A}_{g_n}$) are polynomial in n, while any construction based on *any kind of* monotone formulas such as [26] or [3] is superpolynomial.

The key to the proof of the theorem will be the following lemma:

LEMMA 3 *Let $h : \{0,1\}^n \rightarrow \{0,1\}$ be any monotone function, and suppose that an MSP of size m computes h. Then there exist monotone functions $f : \{0,1\}^{n+1} \rightarrow \{0,1\}$ and $g : \{0,1\}^{n+2} \rightarrow \{0,1\}$ such that: f is maximal Q^2, g is maximal Q^3, both f and g can be computed by MSPs of size $O(m)$; and finally any monotone Boolean circuit computing f (or g) yields immediately a monotone Boolean circuit computing h on any input of Hamming weight less than $n/2$.*

This can be shown by defining the function $\tilde{h}$ on n variables as

$$\tilde{h}(\mathbf{x}) = h(\mathbf{x}) \vee T_n^{n/2}(\mathbf{x}).$$

Then we can construct the claimed functions f, g by introducing two new Boolean variables y, z and defining

$$f(\mathbf{x}, y) = T_3^2(y, \tilde{h}(\mathbf{x}), \tilde{h}^*(\mathbf{x}))$$

and

$$g(\mathbf{x}, y, z) = f(\mathbf{x}, y) \vee (z \wedge (\vee_{i=1}^n x_i)).$$

The claims on the properties of f, g follow from a theorem on the composition of MSP's (see [30]), the well known fact that threshold functions have linear MSP complexity and that a function and its dual have the same MSP complexity (see e.g. [21]).

In [1], the monotone function ODDFACTOR is studied. This function is defined on bipartite graphs on $t = \sqrt{n}$ pairs of vertices, where an input graph can be specified by n Boolean variables, one for each of the potential edges. An input graph is accepted if and only it contains an odd factor, i.e. a subgraph in which every vertex has odd degree (in particular degree > 0). The following result is proved in [1]:

THEOREM 9 *The ODDFACTOR function on n variables can be computed by an MSP over $GF(2)$ of size n, but a monotone Boolean circuit computing the function must have size $n^{\Omega(\log n)}$.*

Now suppose we are given a monotone Boolean circuit family that for any n computes ODDFACTOR on input graphs with less than $n/2$ edges. A simple reduction will show that this family can in fact compute ODDFACTOR in general, at polynomial extra cost. This, together with Theorem 9 and Lemma 3, immediately implies Theorem 8.

Acknowledgments

We thank Serge Fehr, Mathias Fitzi, Anna Gál, Rosario Gennaro, Martin Hirt, Peter Bro Miltersen, and Tal Rabin for interesting discussions and comments.

References

1. L. Babai, A. Gál, J. Kollár, L. Rónyai, T. Szabó, A. Wigderson: *Extremal bipartite graphs and super-polynomial lower bounds for monotone span programs*, Proc. ACM STOC '96, pp. 603–611.
2. D. Beaver, *Foundations of secure interactive computing*, Proc. CRYPTO '91, Springer Verlag LNCS, vol. 576, pp. 377–391.
3. D. Beaver and A. Wool, *Quorum-based multi-party computations*, Proc. EUROCRYPT '98, Springer Verlag LNCS, vol. 1403, pp. 375–390.
4. A. Beimel, *Secure schemes for secret sharing and key distribution*, Ph.D.-thesis, Technion, Haifa, June 1996.
5. J. Benaloh, J. Leichter, *Generalized secret sharing and monotone functions*, Proc. CRYPTO '88, Springer Verlag LNCS, vol. 403, pp. 25–35.
6. M. Ben-Or, S. Goldwasser, A. Wigderson, *Completeness theorems for non-cryptographic fault-tolerant distributed computation*, Proc. ACM STOC '88, pp. 1–10.
7. M. Bertilsson, I. Ingemarsson, *A construction of practical secret sharing schemes using linear block codes*, Proc. AUSCRYPT '92, Springer Verlag LNCS, vol. 718, pp. 67–79.
8. E. F. Brickell, *Some ideal secret sharing schemes*, J. Combin. Maths. & Combin. Comp. 9 (1989), pp. 105–113.
9. R. Canetti, *Studies in secure multi-party computation and applications*, Ph. D. thesis, Weizmann Institute of Science, Rehovot, 1995.
10. R. Canetti, U. Feige, O. Goldreich, M. Naor, *Adaptively secure multi-party computation*, Proc. ACM STOC '96, pp. 639–648.
11. D. Chaum, C. Crépeau, I. Damgård, *Multi-party unconditionally secure protocols*, Proc. ACM STOC '88, pp. 11–19.
12. B. Chor, S. Goldwasser, S. Micali, B. Awerbuch, *Verifiable secret sharing and achieving simultaneity in the presence of faults*, Proc. FOCS '85, pp. 383–395.
13. R. Cramer, I. Damgård, *Zero Knowledge for Finite Field Arithmetic or: Can Zero Knowledge be for Free?*, Proc. CRYPTO'98, Springer Verlag LNCS, vol. 1462, pp. 424–441.
14. R. Cramer, I. Damgård, S. Dziembowski, *On the complexity of verifiable secret sharing and multi-party computation*, Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC '00), Portland, Oregon, May 2000.
15. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt and T. Rabin, *Efficient multi-party computations secure against an adaptive adversary*, Proc. EUROCRYPT '99, Springer Verlag LNCS, vol. 1592, pp. 311–326.
16. C. Crépeau, J. van de Graaf and A. Tapp, *Committed oblivious transfer and private multi-party computation*, proc. CRYPTO '95, Springer Verlag LNCS, vol. 963, pp. 110–123.
17. M. van Dijk, *Secret key sharing and secret key generation*, Ph.D. Thesis, Eindhoven University of Technology, 1997.
18. S. Fehr, *Efficient construction of dual MSP*, manuscript 1999.
19. M. Fitzi, U. Maurer, *Efficient Byzantine agreement secure against general adversaries*, Proc. 12th Int. Symp. on Distributed Computing (DISC '98), Springer Verlag LNCS, vol. 1499, pp. 134–148.
20. A. Gál, *A characterization of span program size and improved lower bounds for monotone span programs*, Proceedings of the 30th ACM Symposium on the Theory of Computing, 1998, pp. 429–437.

21. A. Gál, *Combinatorial methods in Boolean function complexity*, Ph.D.-thesis, University of Chicago, 1995.
22. Z. Galil, S. Haber and M. Yung, *Cryptographic computation: Secure fault-tolerant protocols and the public-key model*, Proc. CRYPTO'87, Springer Verlag LNCS, vol. 293, pp. 135–155.
23. R. Gennaro, *Theory and practice of verifiable secret sharing*, Ph.D. thesis, MIT, 1996.
24. R. Gennaro, M. Rabin, T. Rabin, *Simplified VSS and fast-track multi-party computations with applications to threshold cryptography*, Proc. ACM PODC'98.
25. O. Goldreich, S. Micali and A. Wigderson, *How to play any mental game or a completeness theorem for protocols with honest majority*, Proc. ACM STOC '87, pp. 218–229.
26. M. Hirt, U. Maurer, *Player simulation and general adversary structures in perfect multi-party computation*, Journal of Cryptology, vol. 13, no. 1, pp. 31–60, 2000. (Preliminary version in Proc. ACM PODC'97, pp. 25–34.)
27. M. Ito, A. Saito and T. Nishizeki, *Secret sharing schemes realizing general access structures*, Proc. IEEE GlobeCom '87 Tokyo, pp. 99–102.
28. M. Karchmer, A. Wigderson, *On span programs*, Proc. of Structure in Complexity '93, pp. 102–111.
29. S. Micali and P. Rogaway, *Secure computation*, Manuscript, Preliminary version in Proc. CRYPTO '91, Springer Verlag LNCS, vol. 576, pp. 392–404
30. P. Pudlák, J. Sgall, *Algebraic Models of Computation and Interpolation for Algebraic Proof Systems* Proc. Feasible Arithmetic and Proof Complexity, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, AMS, vol. 39, pp. 279–296, 1996.
31. T. Rabin, M. Ben-Or, *Verifiable secret sharing and multi-party protocols with honest majority*, Proc. ACM STOC '89, pp. 73–85.
32. T. Rabin, *Robust sharing of secrets when the dealer is honest or cheating*, J. ACM, 41(6):1089-1109, November 1994.
33. A. Shamir, *How to share a secret*, Communications of the ACM 22 (1979) 612–613.