Secure Multiparty Computation with Partial Fairness

Amos Beimel*
Department of Computer Science
Ben Gurion University
Be'er Sheva, Israel

Eran Omri†
Department of Computer Science
Bar Ilan University
Ramat Gan, Israel

Ilan Orlov‡
Department of Computer Science
Ben Gurion University
Be'er Sheva, Israel

November 23, 2010

Abstract

A protocol for computing a functionality is secure if an adversary in this protocol cannot cause more harm than in an ideal computation where parties give their inputs to a trusted party which returns the output of the functionality to all parties. In particular, in the ideal model such computation is fair – all parties get the output. Cleve (STOC 1986) proved that, in general, fairness is not possible without an honest majority. To overcome this impossibility, Gordon and Katz (Eurocrypt 2010) suggested a relaxed definition – $1/p$-secure computation – which guarantees partial fairness. For two parties, they construct $1/p$-secure protocols for functionalities for which the size of either their domain or their range is polynomial (in the security parameter). Gordon and Katz ask whether their results can be extended to multiparty protocols.

We study $1/p$-secure protocols in the multiparty setting for general functionalities. Our main result is constructions of $1/p$-secure protocols when the number of parties is constant provided that less than 2/3 of the parties are corrupt. Our protocols require that either (1) the functionality is deterministic and the size of the domain is polynomial (in the security parameter), or (2) the functionality can be randomized and the size of the range is polynomial. If the size of the domain is constant and the functionality is deterministic, then our protocol is efficient even when the number of parties is $O(\log \log n)$ (where n is the security parameter). On the negative side, we show that when the number of parties is super-constant, $1/p$-secure protocols are not possible when the size of the domain is polynomial.

*Supported by ISF grant 938/09 and by the Frankel Center for Computer Science.
†This research was generously supported by the European Research Council as part of the ERC project "LAST".
‡Supported by ISF grant 938/09 and by the Frankel Center for Computer Science.

1 Introduction

A protocol for computing a functionality is secure if an adversary in this protocol cannot cause more harm than in an ideal computation where parties give their inputs to a trusted party which returns the output of the functionality to all parties. This is formalized by requiring that for every adversary in the real world, there is an adversary in the ideal world, called simulator, such that the output of the real-world adversary and the simulator are indistinguishable in polynomial time. Such security can be achieved when there is a majority of honest parties [16]. Secure computation is fair – all parties get the output. Cleve [9] proved that, in general, fairness is not possible without an honest majority.

To overcome the impossibility of [9], Gordon and Katz [22] suggested a relaxed definition – $1/p$-secure computation – which guarantees partial fairness. Informally, a protocol is $1/p$-secure if for every adversary in the real world, there is a simulator running in the ideal world, such that the output of the real-world adversary and the simulator cannot be distinguished with probability greater than $1/p$. For two parties, Gordon and Katz construct $1/p$-secure protocols for functionalities whose size of either their domain or their range is polynomial (in the security parameter). They also give impossibility results when both the domain and range are super-polynomial. Gordon and Katz ask whether their results can be extended to multiparty protocols. We give positive and negative answers to this question.

Previous Results. Cleve [9] proved that any protocol for coin-tossing without an honest majority cannot be fully secure, specifically, if the protocol has r rounds, then it is at most $1/r$-secure. Protocols with partial fairness, under various definitions and assumptions, have been constructed for coin-tossing [9, 10, 24, 4], for contract signing/exchanging secrets [6, 23, 12, 5, 11, 7], and for general functionalities [27, 13, 2, 17, 25, 14, 22]. We next describe the papers that are most relevant to our paper. Moran, Naor, and Segev [24] construct 2-party protocols for coin tossing that are $1/r$-secure (where r is the number of rounds in the protocol). Gordon and Katz [22] define $1/p$-security and construct 2-party $1/p$-secure protocols for every functionality whose size of either the domain or the range of the functionality is polynomial. Finlay, in a previous work [4] we construct multiparty protocols for coin tossing that are $O(1/r)$-secure provided that the fraction of bad parties is slightly larger than half. In particular, our protocol is $O(1/r)$-secure when the number of parties is constant and the fraction of bad parties is less than 2/3.

Gordon et al. [20] showed that complete fairness is possible in the two party case for some functions. Gordon and Katz [19] showed similar results for the multiparty case. The characterization of the functions that can be computed with full fairness without honest majority is open. Completeness for fair computations has been studied in [21]. Specifically, they show a specific function that is complete for fair two-party computation; this function is also complete for $1/p$-secure two-party computation.

1.1 Our Results

We study $1/p$-secure protocols in the multiparty setting. We construct two protocols for general functionalities assuming that the fraction of corrupt parties is less than 2/3. The first protocol is efficient when (1) The number of parties is constant, the functionality is deterministic, and the size of the domain of inputs is at most polynomial in the security parameter, or (2) The number of parties is $O(\log \log n)$ (where n is the security parameter), the functionality is deterministic, and the size of the domain of inputs is constant. The second protocol is efficient when the number of parties is constant, the functionality can be randomized, and the size of the *range* of the functionality is at most polynomial in the security parameter. Our second protocol does not provide correctness, i.e., in a case of premature termination, with probability of $1/\operatorname{poly}(n)$, the remaining active parties output a value which might be inconsistent with their inputs. In contrast, our first protocol provides correctness.

Our protocols combine ideas from the protocols of Gordon and Katz [22] and our paper [4], both of which generalize the protocol of Moran, Naor, and Segev [24]. Specifically, our protocols proceed in rounds, where in each round values are given to subsets of parties. There is a special round $i^\star$ in the protocol. Prior to round $i^\star$, the values given to a subset of parties are values that can be computed from the inputs of the parties in this subset; staring from round $i^\star$ the values are the "correct" output of the functionality. The values given to a subset are secret shared such that only if all parties in the subset cooperate they can reconstruct the value. If in some round many (corrupt) parties have aborted such that there is a majority of honest parties among the active parties, then the set of active parties reconstructs the value given to this set in the previous round.[1] Similar to the protocols of [24, 22, 4], the adversary can cause harm (e.g., bias the output of the functionality) only if it guesses $i^\star$; we show that in our protocols this probability is small and the protocols are $1/p$-secure. The values in our protocols are chosen similar to [22]. The mechanism to secret share the values is similar to [4], however, there are important differences in this sharing, as the sharing mechanism of [4] is not appropriate for $1/p$-secure computations of functionalities which depend on inputs.

To complete the picture, we prove interesting impossibility results. We show that, in general, when the number of parties is super-constant, $1/p$-secure protocols are not possible without honest majority when the size of the domain is polynomial. This impossibility result justifies the fact why in our protocols the number of parties is constant. We also show that, in general, when the number of parties is $\omega(\log n)$, $1/p$-secure protocols are not possible without honest majority even when the size of the domain is 2. The proof of the impossibility result is rather simple and follows from an impossibility result of [22].

Our impossibility results should be contrasted with the coin-tossing protocol of [4] which is an efficient $1/p$-secure protocol even when $m(n)$, the number of parties, is polynomial in the security parameter and the number of bad parties is $m(n)/2 + O(1)$. Our results show that these parameters are not possible for general $1/p$-secure protocols even when the size of the domain of inputs is 2.

Open Problems. In both our impossibility results the size of the range is super-polynomial. It is open if there is an efficient $1/p$-secure protocol when the number of parties is not constant and the size of both the domain and range is polynomial. In addition, the impossibility results do not rule out that the double-exponential dependency on the number of parties can be improved.

The protocols of [22] are private – the adversary cannot learn any information on the inputs of the honest parties (other than the information that it can learn in the ideal world of computing $\mathcal{F}$). The adversary can only bias the output. Our first protocol is not private (that is, the adversary can learn extra information). However, we do not know whether the second protocol is private.[2] It is open if there are general multiparty $1/p$-secure protocols that are also private.

2 Preliminaries

A multi-party protocol with m parties is defined by m interactive probabilistic polynomial-time Turing machines $p_1, \ldots, p_m$. Each Turning machine, called party, has the security parameter 1^n as a joint input and a private input y_j. The computation proceeds in rounds. In each round, the active parties broadcast and receive messages on a common broadcast channel. The number of rounds in the protocol is expressed as some function $r(n)$ in the security parameter (typically, $r(n)$ is bounded by a polynomial). At the end of the protocol, the (honest) parties should hold a common value w (which should be equal to an output of a predefined functionality).

[1] As parties can abort during this reconstruction, they actually reconstruct the value of a subset of this set.

[2] The problem in our protocols is that the adversary can keep one corrupted party active, thus, the adversary can get the output of the honest parties.

In this work we consider a corrupt, static, computationally-bounded (i.e., non-uniform probabilistic polynomial-time) adversary that is allowed to corrupt some subset of parties. That is, before the beginning of the protocol, the adversary corrupts a subset of the parties and may instruct them to deviate from the protocol in an arbitrary way. The adversary has complete access to the internal state of each of the corrupted parties and fully controls the messages that they send throughout the protocol. The honest parties follow the instructions of the protocol.

The parties communicate via a synchronous network, using only a broadcast channel. The adversary is rushing, that is, in each round the adversary hears the messages broadcast by the honest parties before broadcasting the messages of the corrupted parties for this round (thus, broadcast messages of the corrupted parties can depend on the broadcast messages of the honest parties in this round).

Notation. For an integer ℓ, define $[\ell] = \{1, \ldots, \ell\}$. For a set $J \subseteq [m]$, define $Q_J = \{p_j : j \in J\}$. An m-party functionality $\mathcal{F} = \{f_n\}_{n\in\mathbb{N}}$ is a sequence of polynomial-time computable, randomized mappings $f_n : (X_n)^m \to Z_n$, where $X_n = \{0,1\}^{\ell_d(n)}$ and $Z_n = \{0,1\}^{\ell_r(n)}$ are the domain of inputs of each party and the range respectively; $\ell_d, \ell_r : \mathbb{N} \to \mathbb{N}$ are some fixed functions. We denote the size of the domain and the range of $\mathcal{F}$ by $d(n)$ and $g(n)$ respectively, that is, $d(n) = 2^{\ell_d(n)}$ and $g(n) = 2^{\ell_r(n)}$. For a randomized mapping f_n, the assignment $w \leftarrow f_n(x_1, \ldots, x_m)$ denotes the process of computing f_n with the inputs $x_1, \ldots, x_m$ and with uniformly chosen random coins and assigning the output of the computation to w. If $\mathcal{F}$ is deterministic, we sometimes call it a function. We sometime omit n from functions of n (for example, we write d instead of $d(n)$).

2.1 The Real vs. Ideal Paradigm

The security of multiparty computation protocols is defined using the real vs. ideal paradigm. In this paradigm, we consider the real-world model, in which protocols are executed. We then formulate an ideal model for executing the task. This ideal model involves a trusted party whose functionality captures the security requirements from the task. Finally, we show that the real-world protocol "emulates" the ideal-world protocol: For any real-life adversary $\mathcal{A}$ there exists an ideal-model adversary $\mathcal{S}$ (called simulator) such that the global output of an execution of the protocol with $\mathcal{A}$ in the real-world model is distributed similarly to the global output of running $\mathcal{S}$ in the ideal model. In both models there are m parties $p_1, \ldots, p_m$ holding a common input 1^n and private inputs $y_1, \ldots, y_m$ respectively, where $y_j \in X_n$ for $1 \le j \le m$.

The Real Model. Let Π be an m-party protocol computing $\mathcal{F}$. Let $\mathcal{A}$ be a non-uniform probabilistic polynomial time adversary that gets the input y_j of each corrupted party p_j and the auxiliary input aux. Let $\mathrm{REAL}_{\Pi,\mathcal{A}(\mathrm{aux})}(\vec{y}, 1^n)$, where $\vec{y} = (y_1, \ldots, y_m)$, be the random variable consisting of the view of the adversary (i.e., the inputs of the corrupted parties and the messages it got) and the output of the honest parties following an execution of Π.

The Ideal Model. The basic ideal model we consider is a model without abort. Specifically, there is an adversary $\mathcal{S}$ which has corrupted a subset B of the parties. The adversary $\mathcal{S}$ has some auxiliary input aux. An ideal execution for the computing $\mathcal{F}$ proceeds as follows:

Send inputs to trusted party: The honest parties send their inputs to the trusted party. The corrupted parties may either send their received input, or send some other input of the same length (i.e., $x_j \in X_n$) to the trusted party, or abort (by sending a special "$\texttt{abort}_j$" message). Denote by $x_1, \ldots, x_m$ the inputs received by the trusted party. If p_j does not send an input, then the trusted party selects $x_j \in X_n$

with uniform distribution.[3]

Trusted party sends outputs: The trusted party computes $f_n(x_1, \ldots, x_m)$ with uniformly random coins and sends the output to the parties.

Outputs: The honest parties output the value sent by the trusted party, the corrupted parties output nothing, and $\mathcal{S}$ outputs any arbitrary (probabilistic polynomial-time computable) function of its view (its inputs, the output, and the auxiliary input aux).

Let $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}(\mathrm{aux})}(\vec{y}, 1^n)$ be the random variable consisting of the output of the adversary $\mathcal{S}$ in this ideal world execution and the output of the honest parties in the execution.

2.1.1 $1/p$-Indistinguishability and $1/p$-Secure Computation

As explained in the introduction, some ideal functionalities for computing $\mathcal{F}$ cannot be implemented when there is no honest majority. We use $1/p$-secure computation, defined by [22], to capture the divergence from the ideal worlds.

Definition 2.1 ($1/p$-indistinguishability) *A function $\mu(\cdot)$ is* negligible *if for every positive polynomial $q(\cdot)$ and all sufficiently large n it holds that $\mu(n) < 1/q(n)$. A* distribution ensemble $X = \{X_{a,n}\}_{a\in\mathcal{D}_n, n\in\mathbb{N}}$ *is an infinite sequence of random variables indexed by $a \in \mathcal{D}_n$ and $n \in \mathbb{N}$, where $\mathcal{D}_n$ is a domain that might depend on n. For a fixed function $p(n)$, two distribution ensembles $X = \{X_{a,n}\}_{a\in\mathcal{D}_n,n\in\mathbb{N}}$ and $Y = \{Y_{a,n}\}_{a\in\mathcal{D}_n,n\in\mathbb{N}}$ are* computationally $1/p$-indistinguishable*, denoted $X \overset{1/p}{\approx} Y$, if for every non-uniform polynomial-time algorithm D there exists a negligible function $\mu(\cdot)$ such that for every n and every $a \in \mathcal{D}_n$,*

$$\Big|\Pr[D(X_{a,n}) = 1] - \Pr[D(Y_{a,n}) = 1]\Big| \leq \frac{1}{p(n)} + \mu(n).$$

Two distribution ensembles are *computationally indistinguishable*, denoted $X \overset{\mathrm{c}}{\equiv} Y$, if for every $c \in \mathbb{N}$ they are computationally $\frac{1}{n^c}$ -indistinguishable.

We next define the notion of $1/p$-secure computation [22]. The definition uses the standard real/ideal paradigm [15, 8], except that we consider a completely fair ideal model (as typically considered in the setting of honest majority), and require only $1/p$-indistinguishability rather than indistinguishability.

Definition 2.2 ($1/p$-secure computation [22]) *Let $p = p(n)$ be a function. An m-party protocol Π is said to $1/p$-securely compute a functionality $\mathcal{F}$ where there are at most $t(n)$ corrupt parties, if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ in the real model controlling at most $t(n)$ parties, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model, controlling the same parties as $\mathcal{A}$, such that the following two distribution ensembles are computationally $1/p$-indistinguishable*

$$\left\{\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}(\mathrm{aux})}(\vec{y}, 1^n)\right\}_{\mathrm{aux}\in\{0,1\}^*,\vec{y}\in(X_n)^m,n\in\mathbb{N}} \overset{1/p}{\approx} \left\{\mathrm{REAL}_{\Pi,\mathcal{A}(\mathrm{aux})}(\vec{y}, 1^n)\right\}_{\mathrm{aux}\in\{0,1\}^*,\vec{y}\in(X_n)^m,n\in\mathbb{N}}.$$

We next define statistical distance between two random variables and the notion of perfect $1/p$-secure computation, which implies the notion of $1/p$-secure computation.

[3] For the simplicity of the presentation of our protocols, we present a slightly different ideal world than the traditional one. In our model there is no a default input in case of an "abort". However, the protocol can be presented in the traditional model, where a predefined default input is used if a party aborts.

Definition 2.3 (statistical distance) *We define the* statistical distance *between two random variables A and B as the function*

$$\mathrm{SD}\left(A,B\right)=\frac{1}{2}\sum_{\alpha}\Big|\Pr[A=\alpha]-\Pr[B=\alpha]\Big|.$$

Definition 2.4 (perfect $1/p$-secure computation) *An m-party protocol Π is said to perfectly $1/p$-secure compute a functionality $\mathcal{F}$ if for every non-uniform adversary $\mathcal{A}$ in the real model, there exists a polynomial-time adversary $\mathcal{S}$ in the ideal model such that for every $n\in\mathbb{N}$, for every $\vec{y}\in(X_n)^m$, and for every $\mathrm{aux}\in\{0,1\}^*$*

$$\mathrm{SD}\left(\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}(\mathrm{aux})}(\vec{y},1^n),\mathrm{REAL}_{\Pi,\mathcal{A}(\mathrm{aux})}(\vec{y},1^n)\right)\leq\frac{1}{p(n)}.$$

Security with abort and cheat detection is defined in Appendix A. The cryptographic tools we use are described in Appendix B.

3 The Multiparty Secure Protocols

In this section we present our protocols. We start with a protocol that assumes that either the functionality is deterministic and the size of the domain is polynomial, or that the functionality is randomized and both the domain and range of the functionality are polynomial. We then present a modification of the protocol that is $1/p$-secure for (possibly randomized) functionalities if the size of the range is polynomial (even if the size of the domain of $\mathcal{F}$ is not polynomial). The first protocol is more efficient for deterministic functionalities with polynomial-size domain. Furthermore, the first protocol has full correctness, while in the modified protocol, correctness is only guaranteed with probability $1-1/p$.

Formally, we prove the following two theorems.

Theorem 1 *Let $\mathcal{F}=\{f_n:(X_n)^m\rightarrow Z_n\}$ be randomized functionality where the size of domain is $d(n)$ and the size of the range is $g(n)$, and let $p(n)$ be a polynomial. If enhanced trap-door permutations exist, then for any m and t such that $m/2\leq t<2m/3$, and for any polynomial $p(n)$ there is an $r(n)$-round m-party $1/p(n)$-secure protocol computing $\mathcal{F}$ tolerating up to t corrupt parties where $r(n)=p(n)\cdot\left(2\cdot d(n)^m\cdot g(n)\cdot p(n)\right)^{2^t}$, provided that $r(n)$ is bounded by a polynomial in n. If $\mathcal{F}$ is deterministic, then there is a $r(n)$-round $1/p(n)$-secure protocol for $r(n)=p(n)\cdot d(n)^{m\cdot 2^t}$, provided that $r(n)$ is bounded by a polynomial in n.*

Theorem 2 *Let $\mathcal{F}=\{f_n:(X_n)^m\rightarrow Z_n\}$ be randomized functionality where the size of the range $g(n)$ is polynomial in n and m is constant, and let $p(n)$ be a polynomial. If enhanced trap-door permutations exist, then for t such that $m/2\leq t<2m/3$ and for any polynomial $p(n)$ there is an $r(n)$-round m-party $1/p(n)$-secure protocol computing $\mathcal{F}$ tolerating up to t corrupt parties where $r(n)=\left((2p(n))^{2^t+1}\cdot g(n)^{2^t}\right)$.*

Following [24, 4], we present the first protocol in two stages. We first describe in Section 3.1 a protocol with a dealer and then in Section 3.2 present a protocol without this dealer. The goal of presenting the protocol in two stages is to simplify the understanding of the protocol and to enable to prove the protocol in a modular way. In Section 3.3, we present a modification of the protocol which is $1/p$-secure if the size of the range is polynomial (even if the size of the domain of f is not polynomial).

3.1 The Protocol for Polynomial-Size Domain with a Dealer

We consider a network with m parties where at most t of them are corrupt such that $m/2 \leq t \leq 2m/3$. In this section we assume that there is a special trusted on-line dealer, denoted T. This dealer interacts with the parties in rounds, sending messages on private channels. We assume that the dealer knows the set of corrupt parties. In Section 3.2, we show how to remove this dealer and construct a protocol without a dealer.

In our protocol the dealer sends in each round values to subsets of parties; the protocol proceeds with the normal execution as long as at least $t+1$ of the parties are still active. If at some round i, there are at most t active parties, then the active parties reconstruct the value given to them in round $i-1$, output this value, and halt. Following [24], and its follow up works [22, 4], the dealer chooses at random with uniform distribution a special round $i^\star$. Prior to this round the adversary gets no information and if the corrupt parties abort the execution prior to $i^\star$, then they cannot bias the output of the honest parties or cause any harm. After round $i^\star$, the output of the protocol is fixed, and, also in this case the adversary cannot affect the output of the honest parties. The adversary cause harm only if it guesses $i^\star$ and this happens with small probability.

We next give a verbal description of the protocol. This protocol is designed such that the dealer can be removed from it in Section 3.2. A formal description is given in Figure 1. At the beginning of the protocol each party sends its input y_j to the dealer. The corrupted parties may send any values of their choice. Let $x_1, \ldots, x_m$ denote the inputs received by the dealer. If a corrupt party p_j does not send its input, then the dealer sets x_j to be a random value selected uniformly from X_n. In a preprocessing phase, the dealer T selects uniformly at random a special round $i^\star \in [r]$. The dealer computes $w \leftarrow f_n(x_1, \ldots, x_m)$. Then, for every round $1 \leq i < r$ and every $J \subseteq \{1, \ldots, m\}$ such that $m - t \leq |J| \leq t$, the dealer selects an output, denoted σ_J^i, as follows (this output is returned by the parties in $Q_J = \{p_j : j \in J\}$ if the protocol terminates in round $i+1$ and Q_J is the set of the active parties):

CASE I: $1 \leq i < i^\star$. For every $j \in J$ the dealer sets $\widehat{x}_j = x_j$ and for every $j \notin J$ it chooses $\widehat{x}_j$ independently with uniform distribution from the domain X_n; it computes the output $\sigma_J^i \leftarrow f_n(\widehat{x}_1, \ldots, \widehat{x}_m)$.

CASE II: $i^\star \leq i \leq r$. The dealer sets $\sigma_J^i = w$.

The dealer T interacts with the parties in rounds, where in round i, for $1 \leq i \leq r$, there are of three phases:

The peeking phase. The dealer sends to the adversary all the values σ_J^i such that all parties in Q_J are corrupted.

The abort and premature termination phase. The adversary sends to T the identities of the parties that abort in the current round. If there are less than $t+1$ active parties, then T sends σ_J^{i-1} to the active parties, where Q_J is the set of the active parties when parties can also abort during this phase (see exact details in Figure 1). The honest parties return this output and halt.

The main phase. If at least $t+1$ parties are active, T notifies the active parties that the protocol proceeds normally.

If after r rounds, there are at least $t+1$ active parties, T sends w to all active parties and the honest parties output this value.

Example 3.1 *As an example, assume that $m = 5$ and $t = 3$. In this case the dealer computes a value σ_J^i for every set of size 2 or 3. Consider an execution of the protocol where p_1 aborts in round 4 and p_3 and p_4 abort in round 100. In this case, T sends $\sigma_{\{2,5\}}^{99}$ to p_2 and p_5, which return this output.*

Inputs: Each party p_j holds a private input $y_j \in X_n$ and the joint input: the security parameter 1^n, the number of rounds $r = r(n)$, and a bound t on the number of corrupted parties.

Instructions for each honest party p_j: (1) After receiving the "`start`" message, send input y_j to the dealer. (2) If the premature termination step is executed with $i = 1$, then send its input y_j to the dealer. (3) Upon receiving output z from the dealer, output z. (Honest parties do not send any other messages throughout the protocol.)

Instructions for the (trusted) dealer:

The preprocessing phase:

1. Set $D_0 = \emptyset$ and send a "`start`" message to all parties.
2. Receive an input, denoted x_j, from each party p_j. For every p_j that sends an "$\texttt{abort}_j$" message, notify all parties that party p_j aborted, select $x_j \in X_n$ with uniform distribution, and update $D_0 = D_0 \cup \{j\}$.
3. Let $D = D_0$. If $|D| \geq m - t$, go to premature termination with $i = 1$.
4. Set $w \leftarrow f_n(x_1, \ldots, x_m)$ and select $i^\star \in \{1, \ldots, r\}$ with uniform distribution.
5. For each $1 \leq i < i^\star$, for each $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$: for each $j \in J$ set $\widehat{x}_j = x_j$, for each $j \notin J$ select uniformly at random $\widehat{x}_j \in X_n$, and set $\sigma_J^i \leftarrow f_n(\widehat{x}_1, \ldots, \widehat{x}_m)$.
6. For each $i^\star \leq i \leq r$ and for each $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$, set $\sigma_J^i = w$.
7. Send "`proceed`" to all parties.

Interaction rounds: In each round $1 \leq i \leq r$, interact with the parties in three phases:

- **The peeking phase:** For each $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$, if Q_J contains only corrupt parties, send the value σ_J^i to all parties in Q_J.
- **The abort phase:** Upon receiving an "$\texttt{abort}_j$" message from a party p_j, notify all parties that party p_j aborted (ignore all other types of messages) and update $D = D \cup \{j\}$. If $|D| \geq m - t$, go to premature termination step.
- **The main phase:** Send "`proceed`" to all parties.

Premature termination step:

- If $i = 1$, then: Receive an input, denoted $x_j{}'$, from each active party p_j. For every party p_j that sends an "$\texttt{abort}_j$" message, update $D = D \cup \{j\}$ and select $x_j{}' \in X_n$ with uniform distribution. Set $w' \leftarrow f_n(x_1{}', \ldots, x_m{}')$.
- Else, if $i > 1$, then: For each "$\texttt{abort}_j$" message received from a party p_j, update $D = D \cup \{j\}$. Set $w' = \sigma_J^{i-1}$ for $J = [m] \setminus D$.
- Send w' to each party p_j s.t. $j \notin D_0$ and halt.

Normal termination: If the last round of the protocol is completed, send w to to each party p_j s.t. $j \notin D_0$.

Figure 1: Protocol $\mathrm{MPCWithD}_r$.

The formal proof of the $1/p$-security of the protocol appears in Appendix C. We next hint why for deterministic functionalities, any adversary can cause harm in the above protocol by at most $O(d^{O(1)}/r)$, where $d = d(n)$ is the size of the domain of the inputs and the number of parties, i.e., m, is constant. As in the protocols of [24, 22, 4], the adversary can only cause harm by causing the protocol to terminate in round $i^\star$. In our protocol, if in some round there are two values σ^i_J and $\sigma^i_{J'}$ that the adversary can obtain such that $\sigma^i_J \neq \sigma^i_{J'}$, then the adversary can deduce that $i < i^\star$. Furthermore, the adversary might have some auxiliary information on the inputs of the honest parties, thus, the adversary might be able to deduce that a round is not $i^\star$ even if all the values that it gets are equal. However, there are less than 2^t values that the adversary can obtain in each round (i.e., the values of subsets of the t corrupt parties of size at least $m - t$). We will show that for a round i such that $i < i^\star$, the probability that all these values are equal to a fixed value is $1/d^{O(1)}$ for a deterministic function f_n (for a randomized functionality this probability also depends on the size of the range). By [22, Lemma 2], the protocol is $d^{O(1)}/r$-secure.

3.2 Eliminating the Dealer of the Protocol

We eliminate the trusted on-line dealer in a few steps using a few layers of secret-sharing schemes. First, we change the on-line dealer, so that, in each round i, it shares the value σ^i_J of each subset Q_J among the parties of Q_J using a $|J|$-out-of-$|J|$ secret-sharing scheme – called *inner* secret-sharing scheme. As in Protocol $\mathrm{MPCWithD}_r$ described in Figure 1, the adversary is able to obtain information on σ^i_J only if it controls all the parties in Q_J. On the other hand, the honest parties can reconstruct σ^{i-1}_J (without the dealer), where Q_J is the set of active parties containing the honest parties. In the reconstruction, if an active (corrupt) party does not give its share, then it is removed from the set of active parties Q_J. This is possible since in the case of a premature termination an honest majority among the active parties is guaranteed (as further explained below).

Next, we convert the on-line dealer to an off-line dealer. That is, we construct a protocol in which the dealer sends only one message to each party in an initialization stage; the parties interact in rounds using a broadcast channel (without the dealer) and in each round i each party learns its shares of the ith round inner secret-sharing schemes. In each round i, each party p_j learns a share of σ^i_J in a $|J|$-out-of-$|J|$ secret-sharing scheme, for every set Q_J such that $j \in J$ and $m - t \leq |J| \leq t$ (that is, it learns the share of the inner scheme). For this purpose, the dealer computes, in a preprocessing phase, the appropriate shares for the inner secret-sharing scheme. For each round, the shares of each party p_j are then shared in a 2-out-of-2 secret-sharing scheme, where p_j gets one of the two shares (this share is a mask, enabling p_j to privately reconstruct its shares of the appropriate σ^i_J although messages are sent on a broadcast channel). All other parties get shares in a t-out-of-$(m-1)$ Shamir secret-sharing scheme of the other share of the 2-out-of-2 secret-sharing. See Construction B.1 for a formal description. We call the resulting secret-sharing scheme the *outer* scheme.

To prevent corrupt parties from cheating, by say, sending false shares and causing reconstruction of wrong secrets, every message that a party should send during the execution of the protocol is signed in the preprocessing phase (together with the appropriate round number and with the party's index). In addition, the dealer sends a verification key to each of the parties. To conclude, the off-line dealer gives each party the signed shares for the outer secret sharing scheme together with the verification key. A formal description of the functionality of the off-line dealer, called Functionality $\mathrm{MultiShareGen}$, is given in Figure 2.

The protocol with the off-line dealer proceeds in rounds. In round i of the protocol all parties broadcast their (signed) shares in the outer $(t+1)$-out-of-m secret-sharing scheme. Thereafter, each party can unmask the message it receives (with its share in the appropriate 2-out-of-2 secret-sharing scheme) to obtain its shares in the $|J|$-out-of-$|J|$ inner secret-sharing of the values σ^i_J (for the appropriate sets Q_J's to which the party belongs). If a party stops broadcasting messages or broadcasts improperly signs messages, then all

Joint input: The security parameter 1^n, the number of rounds in the protocol $r = r(n)$, a bound t on the number of corrupted parties, and the set of indices of aborted parties D_0.

Private input: Each party p_j, where $j \notin D_0$, has an input $x_j \in X_n$.

Computing default values and signing keys

1. For every $j \in D_0$, select x_j with uniform distribution from X_n.
2. Select $i^\star \in [r]$ with uniform distribution and compute $w \leftarrow f_n(x_1, \ldots, x_m)$.
3. For each $1 \leq i < i^\star$, for each $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$,
   (a) For each $j \in J$, set $\widehat{x}_j = x_j$.
   (b) For each $j \notin J$, select uniformly at random $\widehat{x}_j \in X_n$.
   (c) Set $\sigma^i_J \leftarrow f_n(\widehat{x}_1, \ldots, \widehat{x}_m)$.
4. For each $i^\star \leq i \leq r$ and for each $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$, set $\sigma^i_J = w$.
5. Compute $(K_{\mathrm{sign}}, K_{\mathrm{ver}}) \leftarrow \mathrm{Gen}(1^n)$.

Computing signed shares of the inner secret-sharing scheme

6. For each $i \in \{1, \ldots, r\}$ and for each $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$,
   (a) Create shares of σ^i_J in a $|J|$-out-of-$|J|$ secret-sharing scheme for the parties in Q_J. For each party $p_j \in Q_J$, let $S^{i,J}_j$ be its share of σ^i_J.
   (b) Sign each share $S^{i,J}_j$: compute $R^{i,J}_j \leftarrow (S^{i,J}_j, i, J, j, \mathrm{Sign}((S^{i,J}_j, i, J, j), K_{\mathrm{sign}}))$.

Computing shares of the outer secret-sharing scheme

7. For each $i \in [r]$, for each $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$, and each $j \in J$, share $R^{i,J}_j$ using a $(t+1)$-out-of-m secret-sharing scheme with respect to p_j as defined in Construction B.1: compute one masking share $\mathrm{mask}_j(R^{i,J}_j)$ and $m-1$ complement shares $(\mathrm{comp}_1(R^{i,J}_j), \ldots, \mathrm{comp}_{j-1}(R^{i,J}_j), \mathrm{comp}_{j+1}(R^{i,J}_j), \ldots, \mathrm{comp}_m(R^{i,J}_j))$.

Signing the messages of all parties

8. For every $1 \leq q \leq m$, compute the message $m_{q,i}$ that $p_q \in P$ broadcasts in round i by concatenating (1) q, (2) i, and (3) the complement shares $\mathrm{comp}_q(R^{i,J}_j)$ produced in Step (7) for p_q (for all $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$ and all $j \neq q$ s.t. $j \in J$), and compute $M_{q,i} \leftarrow (m_{q,i}, \mathrm{Sign}(m_{q,i}, K_{\mathrm{sign}}))$.

Outputs: Each party p_j such that $j \notin D_0$ receives

- The verification key K_{ver}.
- The messages $M_{j,1}, \ldots, M_{j,r}$ that p_j broadcasts during the protocol.
- p_j's private masks $\mathrm{mask}_j(R^{i,J}_j)$ produced in Step (7), for each $1 \leq i \leq r$ and each $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$ and $j \in J$.

Figure 2: The initialization functionality $\mathrm{MultiShareGen}_r$.

other parties consider it as aborted. If $m - t$ or more parties abort, the remaining parties reconstruct the value of the set that contains all of them, i.e., σ_J^{i-1}. In the special case of premature termination already in the first round, the remaining active parties engage in a fully secure protocol (with honest majority) to compute f_n.

The use of the outer secret-sharing scheme with threshold $t+1$ plays a crucial role in eliminating the on-line dealer. On the one hand, it guarantees that an adversary, corrupting at most t parties, cannot reconstruct the shares of round i before round i. On the other hand, at least $m - t$ parties must abort to prevent the reconstruction of the outer secret-sharing scheme (this is why we cannot proceed after $m-t$ parties aborted). Furthermore, since $t \leq 2m/3$, when at least $m - t$ corrupt parties aborted, there is an honest majority. To see this, assume that at least $m - t$ corrupt parties aborted. Thus, at most $t - (m - t) = 2t - m$ corrupt parties are active. There are $m - t$ honest parties (which are obviously active), therefore, as $2t-m < m - t$ (since $t < 2m/3$), an honest majority is achieved when $m - t$ parties abort. In this case we can execute a protocol with full security for the reconstruction.

Finally, we replace the off-line dealer by using a secure-with-abort and cheat-detection protocol computing the functionality computed by the dealer, that is, Functionality $\mathrm{MultiShareGen}_r$. Obtaining the outputs of this computation, an adversary is unable to infer any information regarding the input of honest parties or the output of the protocol (since it gets t shares of a $(t+1)$-out-of-m secret-sharing scheme). The adversary, however, can prevent the execution, at the price of at least one corrupt party being detected cheating by all other parties. In such an event, the remaining parties will start over without the detected cheating party. This goes on either until the protocol succeeds or there is an honest majority and a fully secure protocol computing f_n is executed.

A formal description of the protocol appears in Figure 3. The reconstruction functionality used in this protocol (when at least $m-t$ parties aborted) appears in Figure 4. The details of how to construct a protocol secure-with-abort and cheat-detection with $O(1)$ rounds are given in [4].

Comparison with the multiparty coin-tossing protocol of [4]. Our protocol combines ideas from the protocols of [22, 4]. However, there are some important differences between our protocol and the protocol of [4]. In the coin-tossing protocol of [4], the bits σ_J^i are shared using a threshold scheme where the threshold is smaller than the size of the set Q_J. This means that a proper subset of Q_J containing corrupt parties can reconstruct σ_J^i. In coin-tossing this is not a problem since there are no inputs. However, when computing functionalities with inputs, such σ_J^i might reveal information on the inputs of honest parties in Q_J, and we share σ_J^i with threshold $|Q_J|$. As a result, we use more sets Q_J than in [4] and the bias of the protocol is increased (put differently, to keep the same security, we need to increase the number of rounds in the protocol). For example, the protocol of [4] has small bias when there are polynomially many parties and $t = m/2$. Our protocol is efficient only when there are constant number of parties. As explained in Section 4, this difference is inherent as a protocol for general functionalities with polynomially many parties and $t = m/2$ cannot have a small bias.

3.3 A $1/p$-Secure Protocol for Polynomial Range

Using an idea of [22], we modify our protocol such that it will have a small bias when the size of the range of the functionality $\mathcal{F}$ is polynomially bounded (even if $\mathcal{F}$ is randomized and has a big domain of inputs). The only modification is the way that each σ_J^i is chosen prior to round $i^\star$: with probability $1/(2p)$ we choose σ_J^i as a random value in the range of f_n and with probability $1 - 1/(2p)$ we choose it as in Figure 2. Formally, in the model with the dealer, in the preprocessing phase of $\mathrm{MPCWithD}_r$ described in Figure 1, we replace Step (5) with the following step:

- For each $i \in \{1, \ldots, i^\star - 1\}$ and for each $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$,

Inputs: Each party p_j holds the private input $y_j \in X_n$ and the joint input: the security parameter 1^n, the number of rounds in the protocol $r = r(n)$, and a bound t on the number of corrupted parties.

Preliminary phase:

1. $D_0 = \emptyset$
2. If $|D_0| < m - t$,
   (a) The parties in $\{p_j : j \in [m] \setminus D_0\}$ execute a secure-with-abort and cheat-detection protocol computing Functionality $\mathrm{MultiShareGen}_r$. Each honest party p_j inputs y_j as its input for the functionality.
   (b) If a party p_j aborts, that is, the output of the honest parties is "$\texttt{abort}_j$", then, set $D_0 = D_0 \cup \{j\}$, chose x_j uniformly at random from x_j, and goto Step (2).
   (c) Else (no party has aborted), denote $D = D_0$ and proceed to the first round.
3. Otherwise ($|D_0| \geq m - t$), the premature termination is executed with $i = 1$.

In each round $i = 1, \ldots, r$ do:

4. Each party p_j broadcasts $M_{j,i}$ (containing its shares in the outer secret-sharing scheme).
5. For every p_j s.t. $\mathrm{Ver}(M_{j,i}, K_{\mathrm{ver}}) = 0$ or if p_j broadcasts an invalid or no message, then all parties mark p_j as inactive, i.e., set $D = D \cup \{j\}$. If $|D| \geq m - t$, premature termination is executed.

Premature termination step

6. If $i = 1$, the active parties use a multiparty secure protocol (with full security) to compute f_n: Each honest party inputs y_j and the input of each inactive party is chosen uniformly at random from X_n. The active parties output the result, and halt.
7. Otherwise,
   (a) Each party p_j reconstructs $R_j^{i-1,J}$, the signed share of the inner secret-sharing scheme produced in Step (6) of Functionality $\mathrm{MultiShareGen}_r$, for each $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$ and $j \in J$.
   (b) The active parties execute a secure multiparty protocol with an honest majority to compute Functionality $\mathrm{Reconstruction}$, where the input of each party p_j is $R_j^{i-1,J}$ for every $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$ and $j \in J$.
   (c) The active parties output the output of this protocol, and halt.

At the end of round r:

8. Each active party p_j broadcasts the signed shares $R_j^{r,J}$ for each J such that $j \in J$.
9. Let $J \subseteq [m] \setminus D$ be the lexicographical first set such that all the parties in Q_J broadcast properly signed shares $R_j^{r,J}$. Each active party reconstructs the value σ_J^r, outputs σ_J^r, and halts.

Figure 3: The m-party protocol MPC_r for computing $\mathcal{F}$.

Joint Input: The round number i, the indices of inactive parties D, a bound t on the number of corrupted parties, and the verification key, K_{ver}.

Private Input of p_j: A set of signed shares $R_j^{i-1,J}$ for each $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$ and $j \in J$.

Computation:

1. For each p_j, if p_j's input is not appropriately signed or malformed, then $D = D \cup \{j\}$.
2. Set $J = [m] \setminus D$.
3. Reconstruct σ_J^{i-1} from the shares of all the parties in Q_J.

Outputs: All parties receive the value σ_J^{i-1} (as their output).

Figure 4: Functionality $\mathrm{Reconstruction}$ for reconstructing the output in the premature termination step.

- with probability $1/(2p)$, select uniformly at random $z_J^i \in Z_n$ and set $\sigma_J^i = z_J^i$.
- with the remaining probability $1 - 1/(2p)$,
  1. For every $j \notin J$ select uniformly at random $\widehat{x}_j \in X_n$ and for each $j \in J$, set $\widehat{x}_j = x_j$.
  2. Compute $\sigma_J^i \leftarrow f_n(\widehat{x}_1, \ldots, \widehat{x}_m)$.

Similarly, in the protocol without the dealer, Protocol MPC_r, we replace Step (3) in $\mathrm{MultiShareGen}_r$ (described in Figure 2) with the above step. Denote the resulting protocols with and without the dealer models by $\mathrm{MPCWithDForRange}$ and $\mathrm{MPCForRange}_r$, respectively.

The idea why this change improves the protocol is that now the probability that all values held by the adversary are equal prior to round $i^\star$ is bigger, thus, the probability that the adversary guesses $i^\star$ is smaller. This modification, however, can cause the honest parties to output a value that is not possible given their inputs, and, in general, we cannot simulate the case (which happens with probability $1/(2p)$) when the output is chosen with uniform distribution from the range.

4 Impossibility of $1/p$-secure Computation with Non-Constant Number of Parties

For deterministic functions, our protocol is efficient when the number of parties m is constant and the size of the domain or range is polynomial (in the security parameter n) or when the number of parties is $O(\log \log n)$ and the size of the domain is constant. We next show that, in general, there is no efficient protocol when the number of parties is $m(n) = \omega(1)$ and the size of the domain is polynomial and when $m(n) = \omega(\log n)$ and the size of the domain of each party is 2. This is done using the following impossibility result of Gordon and Katz [22].

Theorem 3 ([22]) *For every $\ell(n) = \omega(\log n)$, there exists a deterministic 2-party functionality $\mathcal{F}$ with domain and range $\{0,1\}^{\ell(n)}$ that cannot be $1/p$-securely computed for $p \geq 2 + 1/\operatorname{poly}(n)$.*

We next state and prove our impossibility results.

Theorem 4 *For every $m(n) = \omega(\log n)$, there exists a deterministic $m(n)$-party functionality $\mathcal{F}'$ with domain $\{0,1\}$ that cannot be $1/p$-securely computed for $p \geq 2 + 1/\operatorname{poly}(n)$ without an honest majority.*

Proof: Let $\ell(n) = m(n)/2$ (for simplicity, assume $m(n)$ is even). Let $\mathcal{F} = \{f_n\}_{n\in\mathbb{N}}$ be the functionality guaranteed in Theorem 3 for $\ell(n)$. Define an $m(n)$-party deterministic functionality $\mathcal{F}' = \{f'_n\}_{n\in\mathbb{N}}$, where in f'_n party p_j gets the jth bit of the inputs of f_n and the outputs of f_n and f'_n are equal Assume that $\mathcal{F}'$ can be $1/p$-securely computed by a protocol Π' assuming that $t(n) = m(n)/2$ parties can be corrupted. This implies a $1/p$-secure protocol Π for $\mathcal{F}$ with two parties, where the first party simulates the first $t(n)$ parties in Π' and the second party simulates the last $t(n)$ parties. The $1/p$-security of Π is implied by the fact that any adversary $\mathcal{A}$ for the protocol Π can be transformed into an adversary $\mathcal{A}'$ for Π' controlling $m(n)/2 = t(n)$ parties; as $\mathcal{A}'$ cannot violate the $1/p$-security of Π', the adversary $\mathcal{A}$ cannot violate the $1/p$-security of Π. □

Theorem 5 *For every $m(n) = \omega(1)$, there exists a deterministic $m(n)$-party functionality $\mathcal{F}''$ with domain $\{0,1\}^{\log n}$ that cannot be $1/p$-securely computed for $p \geq 2 + 1/\operatorname{poly}(n)$ without an honest majority.*

Proof: Let $\ell(m) = 0.5m(n)\log n$ and let $\mathcal{F} = \{f_n\}_{n\in\mathbb{N}}$ be the functionality guaranteed in Theorem 3 for $\ell(m)$. We divide the $2\ell(n)$ bits of the inputs of f_n into $m(n)$ blocks of length $\log n$. Define an $m(n)$-party deterministic functionality $\mathcal{F}'' = \{f''_n\}_{n\in\mathbb{N}}$, where in f''_n party p_j gets the jth block of the inputs of f_n and the outputs of f_n and f''_n are equal. As in the proof of Theorem 4, a $1/p$-secure protocol for $\mathcal{F}''$ implies a $1/p$-secure protocol for $\mathcal{F}$ contradicting Theorem 3. □

The above impossibility results should be contrasted with the coin-tossing protocol of [4] which is an efficient $1/p$-secure protocol even when m is polynomial in the security parameter and the number of bad parties is $m(n)/2 + O(1)$. Notice that in both our impossibility results the size of the range is super-polynomial (as we consider the model where all parties get the same output). It is open if there is an efficient $1/p$-secure protocol when the number of parties is not constant and the size of both the domain and range is polynomial.

References

[1] Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In S. Vadhan, editor, *Proc. of the Fourth Theory of Cryptography Conference – TCC 2006*, volume 4392 of *Lecture Notes in Computer Science*, pages 137–156. Springer-Verlag, 2007.

[2] D. Beaver and S. Goldwasser. Multiparty computation with faulty majority. In *Proc. of the 30th IEEE Symp. on Foundations of Computer Science*, pages 468–473, 1989.

[3] A. Beimel, E. Omri, and I. Orlov. Protocols for multiparty coin toss with dishonest majority. Full version of [4].

[4] A. Beimel, E. Omri, and I. Orlov. Protocols for multiparty coin toss with dishonest majority. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 538–557. Springer-Verlag, 2010.

[5] M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest. A fair protocol for signing contracts. In *Proceedings of the 12th Colloquium on Automata, Languages and Programming*, pages 43–52. Springer-Verlag, 1985.

[6] M. Blum. How to exchange (secret) keys. *ACM Trans. Comput. Syst.*, 1(2):175–193, 1983.

[7] D. Boneh and M. Naor. Timed commitments. In M. Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 236–254. Springer-Verlag, 2000.

[8] R. Canetti. Security and composition of multiparty cryptographic protocols. *J. of Cryptology*, 13(1):143–202, 2000.

[9] R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proc. of the 18th STOC*, pages 364–369, 1986.

[10] R. Cleve. Controlled gradual disclosure schemes for random bits and their applications. In G. Brassard, editor, *Advances in Cryptology – CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 573–588. Springer-Verlag, 1990.

[11] I. Damgård. Practical and provably secure release of a secret and exchange of signatures. *J. of Cryptology*, 8(4):201–222, 1995.

[12] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *CACM*, 28(6):637–647, 1985.

[13] Z. Galil, S. Haber, and M. Yung. Cryptographic computation: Secure fault-tolerant protocols and the public-key model. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 135–155. Springer-Verlag, 1988.

[14] J. A. Garay, P. D. MacKenzie, M. Prabhakaran, and K. Yang. Resource fairness and composability of cryptographic protocols. In S. Halevi and T. Rabin, editors, *Proc. of the Third Theory of Cryptography Conference – TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 404–428. Springer-Verlag, 2006.

[15] O. Goldreich. *Foundations of Cryptography, Voume II Basic Applications*. Cambridge University Press, 2004.

[16] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of the 19th ACM Symp. on the Theory of Computing*, pages 218–229, 1987.

[17] S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology – CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 77–93. Springer-Verlag, 1991.

[18] S. Goldwasser and Y. Lindell. Secure computation without agreement. In *DISC '02: Proceedings of the 16th International Conference on Distributed Computing*, pages 17–32, London, UK, 2002. Springer-Verlag.

[19] D. Gordon and J. Katz. Complete fairness in multi-party computation without an honest majority. In *Proc. of the Sixth Theory of Cryptography Conference – TCC 2009*, pages 19–35, Berlin, Heidelberg, 2009. Springer-Verlag.

[20] S. D. Gordon, C. Hazay, J. Katz, and Y. Lindell. Complete fairness in secure two-party computation. In *Proc. of the 40th ACM Symp. on the Theory of Computing*, pages 413–422, 2008.

[21] S. D. Gordon, Y. Ishai, T. Moran, R. Ostrovsky, and A. Sahai. On complete primitives for fairness. In D. Micciancio, editor, *Proc. of the Seventh Theory of Cryptography Conference – TCC 2010*, volume 5978 of *Lecture Notes in Computer Science*, pages 91–108. Springer-Verlag, 2010.

[22] S. D. Gordon and J. Katz. Partial fairness in secure two-party computation. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 157–176. Springer-Verlag, 2010.

[23] M. Luby, S. Micali, and C. Rackoff. How to simultaneously exchange a secret bit by flipping a symmetrically-biased coin. In *Proc. of the 24th IEEE Symp. on Foundations of Computer Science*, pages 11–21, 1983.

[24] T. Moran, M. Naor, and G. Segev. An optimally fair coin toss. In *Proc. of the Sixth Theory of Cryptography Conference – TCC 2009*, pages 1–18, 2009.

[25] B. Pinkas. Fair secure two-party computation. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 87–105. Springer-Verlag, 2003.

[26] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[27] A. C. Yao. How to generate and exchange secrets. In *Proc. of the 27th IEEE Symp. on Foundations of Computer Science*, pages 162–167, 1986.

A Security with Abort and Cheat Detection

We next present a definition of secure multiparty computation that is more stringent than standard definitions of secure computation with abort. This definition extends the definition for secure computation as given by Aumann and Lindell [1]. Roughly speaking, the definition requires that one of two events is possible: (1) The protocol terminates normally, and *all* parties receive their outputs, or (2) Corrupted parties deviate from the prescribed protocol; in this case the adversary obtains the outputs of the corrupted parties (but nothing else), and all honest parties are given an identity of one party that has aborted. The formal definition uses the real vs. ideal paradigm as discussed in Section 2.1. We next describe the appropriate ideal model.

Execution in the ideal model. Let $B \subseteq [m]$ denote the set of indices of corrupted parties controlled by an adversary $\mathcal{A}$. The adversary $\mathcal{A}$ receives an auxiliary input denoted aux. An ideal execution proceeds as follows:

Send inputs to trusted party: The honest parties send their inputs to the trusted party. The corrupted parties may either send their received input, or send some other input of the same length (i.e., $x_j \in X_n$) to the trusted party, or abort (by sending a special "$\texttt{abort}_j$" message). Denote by $x_1, \ldots, x_m$ the inputs received by the trusted party. If the trusted party receives an "$\texttt{abort}_j$" message, then it sends "$\texttt{abort}_j$" to all honest parties and terminates (if it received "$\texttt{abort}_j$" from more than one j, then it uses the minimal such j).

Trusted party sends outputs to adversary: The trusted party computes $w \leftarrow f_n(x_1, \ldots, x_m)$ and sends the output w to the adversary.

Adversary instructs the trusted party to continue or halt: $\mathcal{A}$ sends either a "$\texttt{continue}$" message or "$\texttt{abort}_j$" to the trusted party for some corrupt party p_j, i.e., $j \in B$. If it sends a "$\texttt{continue}$" message, the trusted party sends w to all honest parties. Otherwise, if the adversary sends "$\texttt{abort}_j$", then the trusted party sends "$\texttt{abort}_j$" to all honest parties.

Outputs: An honest party always outputs the value w it obtained from the trusted party. The corrupted parties output nothing. The adversary $\mathcal{A}$ outputs any (probabilistic polynomial-time computable) function of the auxiliary input aux, the inputs of the corrupt parties, and the value w obtained from the trusted party.

We let $\mathrm{IDEAL}^{\mathrm{CD}}_{\mathcal{F},\mathcal{S}(\mathrm{aux})}(\vec{y},1^n)$ and $\mathrm{REAL}_{\Pi,A(\mathrm{aux})}(\vec{y},1^n)$ be defined as in Section 2.1 (where in this case $\mathrm{IDEAL}^{\mathrm{CD}}_{\mathcal{F},\mathcal{S}(\mathrm{aux})}(\vec{y},1^n)$ refers to the above execution with cheat-detection of $\mathcal{F}$). This ideal model is different from that of [15] in that in the case of an "abort", the honest parties get output "$\mathtt{abort}_j$" and not a $\perp$ symbol. This means that the honest parties *know* an identity of a corrupted party that causes the abort. This cheat-detection is achieved by most multiparty protocols, including that of [16], but not all (e.g., the protocol of [18] does not meet this requirement). Using this notation we define secure computation with abort and cheat-detection.

Definition A.1 (security-with-abort and cheat-detection) *Let $\mathcal{F}$ and Π be as in Definition 2.2. A protocol Π is said to* `securely compute` *$\mathcal{F}$ against at most $t(n)$ corrupt parties with abort and cheat-detection if for every non-uniform polynomial-time adversary $\mathcal{A}$ in the real model controlling at most $t(n)$ parties, there exists a non-uniform polynomial-time adversary $\mathcal{S}$ in the ideal model controlling the same parties, such that*

$$\left\{\mathrm{IDEAL}^{\mathrm{CD}}_{\mathcal{F},\mathcal{S}(\mathrm{aux})}(\vec{y},1^n)\right\}_{\mathrm{aux}\in\{0,1\}^*,\vec{y}\in(X_n)^m,n\in\mathbb{N}} \quad \stackrel{c}{\equiv} \quad \left\{\mathrm{REAL}_{\Pi,A(\mathrm{aux})}(\vec{y},1^n)\right\}_{\mathrm{aux}\in\{0,1\}^*,\vec{y}\in(X_n)^m,n\in\mathbb{N}}.$$

B Cryptographic Tools

Signature Schemes. Informally, a signature on a message proves that the message was created by its presumed sender, and its content was not altered. A signature scheme is a triple $(\mathrm{Gen},\mathrm{Sign},\mathrm{Ver})$ containing the key generation algorithm Gen, which outputs a pair of keys, the signing key K_S and the verification key K_v, the signing algorithm Sign, and the verifying algorithm Ver. We assume that it is infeasible to produce signatures without holding the signing key. For formal definition see [15].

Secret Sharing Schemes. An α-out-of-m secret-sharing scheme is a mechanism for sharing data among a set of parties such that every set of size α can reconstruct the secret, while any smaller set knows nothing about the secret. In this paper, we use two schemes: the XOR-based m-out-of-m scheme (i.e., in this scheme $\alpha = m$) and Shamir's α-out-of-m secret-sharing scheme [26] which is used when $\alpha < m$. In both schemes, for every $\alpha - 1$ parties, the shares of these parties are uniformly distributed and independent of the secret. Furthermore, given such $\alpha - 1$ shares and a secret s, one can *efficiently* complete them to m shares of the secret s.

In our protocols we sometimes require that a single party learns the value of a secret that is shared among all parties. Since all messages are sent over a broadcast channel, we use two layers of secret sharing to obtain the above requirements as described below.

Construction B.1 (secret sharing with respect to a certain party) *Let s be a secret taken from some finite field $\mathbb{F}$. We share s among m parties* with respect to *a (special) party p_j in an α-out-of-m secret-sharing scheme as follows:*

1. *Choose shares $(s^{(1)},s^{(2)})$ of the secret s in a two-out-of-two secret-sharing scheme (that is, select $s^{(1)} \in \mathbb{F}$ uniformly at random and compute $s^{(2)} = s - s^{(1)}$). Denote these shares by $\mathrm{mask}_j(s)$ and $\mathrm{comp}(s)$, respectively.*
2. *Compute shares $(\lambda^{(1)},\ldots,\lambda^{(j-1)},\lambda^{(j+1)},\ldots,\lambda^{(m)})$ of the secret $\mathrm{comp}(s)$ in an $(\alpha-1)$-out-of-$(m-1)$ Shamir's secret-sharing scheme. For each $\ell \neq j$, denote $\mathrm{comp}_\ell(s) = \lambda^{(\ell)}$.*

Output:

- *The share of party p_j is $\mathrm{mask}_j(s)$. We call this share "p_j's* masking *share".*

- *The share of each party p_ℓ, where $\ell \neq j$, is $\mathrm{comp}_\ell(s)$. We call this share "p_ℓ's* complement *share".*

In the above scheme, we share the secret s among the parties in P in an α-out-of-m secret-sharing scheme where only sets of size α that contain p_j can reconstruct the secret. In this construction, for every $\beta < \alpha$ parties, the shares of these parties are uniformly distributed and independent of the secret. Furthermore, given such $\beta < \alpha$ shares and a secret s, one can *efficiently* complete them to m shares of the secret s. In addition, given β shares and a secret s, one can *efficiently* select uniformly at random a vector of shares competing the β shares to m shares of s.

C Proof of $1/p$-Security of the Protocols with a Dealer

In this section we prove that our protocols described in Section 3 that assume an trusted dealer are perfect $1/\operatorname{poly}$-secure implementations of the ideal functionality $\mathcal{F}$. We start by presenting in Appendix C.1 a simulator for Protocol $\mathrm{MPCWithD}_r$. In Appendix C.2, we prove the correctness of the simulation by showing the the global output in the ideal-world is distributed within $1/\operatorname{poly}$ statistical distance from the global output in the real-world. In Appendix C.3, we describe the required modifications to the simulator for the protocol for $\mathcal{F}$ that has a polynomial-size range, and argue that the modified simulation is correct.

C.1 The Simulator for Protocol $\mathrm{MPCWithD}_r$

We next present a simulator $\mathcal{S}_T$ for Protocol $\mathrm{MPCWithD}_r$, described in Figure 1. Let B be the set of indices of corrupted parties in the execution.

The simulator $\mathcal{S}_T$ invokes $\mathcal{A}$ on the set of inputs $\{y_j : j \in B\}$, the security parameter 1^n, and the auxiliary input aux, playing the role of the trusted dealer in the interaction with $\mathcal{A}$.

Simulating the preprocessing phase:

1. $D_0 = \emptyset$.
2. The simulator $\mathcal{S}_T$ sends a "`start`" message to all corrupt parties.
3. $\mathcal{S}_T$ receives a set of inputs $\{x_j : j \in B\}$ that $\mathcal{A}$ submits to the computation of the dealer. If $\mathcal{A}$ does not submit an input on behalf of p_j, i.e., $\mathcal{A}$ sends an "$\texttt{abort}_j$" message, then, the simulator $\mathcal{S}_T$ notifies all corrupted parties that party p_j aborted and updates $D_0 = D_0 \cup \{j\}$.
4. $\mathcal{S}_T$ sets $D = D_0$. If $|D| \geq m - t$, the simulator sets $i = 1$ and proceeds to simulate the premature termination step.
5. $\mathcal{S}_T$ selects $i^\star \in \{1, \ldots, r\}$ with uniform distribution.
6. For each $i \in \{1, \ldots, i^\star - 1\}$ and for each $J \subseteq B \setminus D_0$ s.t. $m - t \leq |J| \leq t$ do
   (a) For each $j \in [m]$, if $j \in J$, then $\mathcal{S}_T$ sets $\widehat{x}_j = x_j$, else, $\mathcal{S}_T$ selects uniformly at random $\widehat{x}_j \in X_n$.
   (b) $\mathcal{S}_T$ sets $\sigma_J^i \leftarrow f_n(\widehat{x}_1, \ldots, \widehat{x}_m)$.
7. The simulator $\mathcal{S}_T$ sends "`proceed`" to all corrupt parties.

Simulating interaction rounds: In each round $1 \leq i \leq r$, the simulator $\mathcal{S}_T$ interacts in three phases with the parties $\{p_j : j \in B \setminus D_0\}$, i.e., the corrupt parties which are active so far:

- **The peeking phase:**
  - If $i = i^\star$, the simulator $\mathcal{S}_T$ sends the set of inputs $\{x_j : j \in B \setminus D_0\}$ to the trusted party computing $\mathcal{F}$ and receives w_S.

– For each $J \subseteq B \setminus D_0$ s.t. $m - t \leq |J| \leq t$ do
  1. If $i \in \{1, \ldots, i^\star - 1\}$, the simulator $\mathcal{S}_T$ sends the value σ_J^i (prepared in the simulation of the preprocessing phase) to all parties in Q_J (i.e., to the adversary).
  2. Else, if $i \in \{i^\star, \ldots, r\}$, $\mathcal{S}_T$ sends the value w_S to all parties in Q_J (i.e., to the adversary).

- **The abort phase:** Upon receiving an "$\texttt{abort}_j$" message from a party p_j,
  1. $\mathcal{S}_T$ notifies all corrupted parties that party p_j aborted.
  2. $\mathcal{S}_T$ updates $D = D \cup \{j\}$.
  3. If at least $m - t$ parties have aborted so far, that is $|D| > m - t$, the simulator $\mathcal{S}_T$ proceeds to simulate the premature termination step.
- **The main phase:** $\mathcal{S}_T$ sends "$\texttt{proceed}$" to all corrupt parties.

Simulating the premature termination step:

- If the premature termination step occurred in round $i = 1$,
  – The simulator $\mathcal{S}_T$ receives a set of inputs $\{x_j{}' : j \in B \setminus D\}$ that $\mathcal{A}$ submits to the computation of the dealer.
  If $\mathcal{A}$ does not submit an input on behalf of p_j, i.e., sends an "$\texttt{abort}_j$" message, then, the simulator $\mathcal{S}$ notifies all corrupted parties that party p_j aborted and updates $D = D \cup \{j\}$.
  – The simulator $\mathcal{S}_T$ sends the set of inputs $\{x_j{}' : j \in B \setminus D\}$ to the dealer and receives w_S.
- If the premature termination step occurred in round $1 < i < i^\star$,
  1. Upon receiving an "$\texttt{abort}_j$" message from a party p_j, the simulator $\mathcal{S}_T$ updates $D = D \cup \{j\}$.
  2. The simulator $\mathcal{S}_T$ sends the set of inputs $\{x_j : j \in B \setminus D\}$ to the trusted party computing $\mathcal{F}$ and receives w_S.
- ($\diamond$ If the premature termination step occurred in round $i^\star \leq i \leq r$, then $\mathcal{S}_T$ already has w_S $\diamond$)
- $\mathcal{S}_T$ sends the value w_S to each party in $\{p_j : j \in B \setminus D_0\}$.

Simulating normal termination: If the last round of the protocol is completed, then $\mathcal{S}_T$ sends w_S to each party in $\{p_j : j \in B \setminus D_0\}$.

At the end of the interaction with $\mathcal{A}$, the simulator will output the sequence of messages exchanged between the simulator and the corrupted parties.

C.2 Proof of the Correctness of the Simulation for $\mathrm{MPCWithD}_r$

In order to prove the correctness of the simulation described in Appendix C.1, we consider the two random variables from Section 2.1, both of the form (V, C), where V describes a possible view of $\mathcal{A}$, and C describes a possible output of the honest parties (i.e., $C \in Z_n$). The first random variable $\mathrm{REAL}_{\mathrm{MPCWithD}_r, \mathcal{A}(\mathrm{aux})}(\vec{y}, 1^n)$ describes the real world – an execution of Protocol $\mathrm{MPCWithD}$, where V describes the view of the adversary $\mathcal{A}$ in this execution, and C is the output of the honest parties in this execution. The second random variable $\mathrm{IDEAL}_{\mathcal{F}, \mathcal{S}_T(\mathrm{aux})}(\vec{y}, 1^n)$ describes the ideal world – an execution with the trusted party computing $\mathcal{F}$ (this trusted party is denoted by $T_{\mathcal{F}}$), where V describes the output of the simulator $\mathcal{S}_T$ in this execution, and C is the output of the honest parties in this execution. For the rest of this section, we simplify notations and denote the above two random variables by $\mathrm{REAL} = (V_{\mathrm{REAL}}, C_{\mathrm{REAL}})$ and $\mathrm{IDEAL} = (V_{\mathrm{IDEAL}}, C_{\mathrm{IDEAL}})$ respectively.

We consider the probability of a given pair (v, c) according to the two different random variables. We compare the two following probabilities: (1) The probability that v is the view of the adversary $\mathcal{A}$ in an execution of Protocol $\mathrm{MPCWithD}_r$ and c is the output of the honest parties in this execution, where the probability is taken over the random coins of the dealer T. (2) The probability that v is the output of the simulator $\mathcal{S}_T$ in an ideal-world execution with the trusted party $T_{\mathcal{F}}$ and c is the output of the honest parties in this execution, where the probability is taken over the random coins of the simulator $\mathcal{S}_T$ and the random coins of the ideal-world trusted party $T_{\mathcal{F}}$.

In Lemma C.3 we prove the correctness of the simulation by showing that the two random variables are within statistical distance $1/\operatorname{poly}$. For the proof of the lemma we need the following claim from [22].

Claim C.1 ([22, Lemma 2]) *Let $\mathcal{A}$ be an adversary in Protocol $\mathrm{MPCWithD}_r$ and let $x_1, \ldots, x_m$ be a set of inputs. Assume that for every possible output w obtained by the dealer using this set of inputs the probability that in a round $i < i^\star$ all the values that the adversary sees are equal to w is at least α. Then, the probability that $\mathcal{A}$ guesses $i^\star$ (i.e., causes premature termination in round $i^\star$) is at most $1/\alpha r$.*

As the adversary might have some auxiliary information on the inputs of the honest parties and know the value of $f_n(x_1, \ldots, x_m)$, the adversary might be able to deduce that a round is not $i^\star$ if not all the values that it gets are equal to this value (or a possible value for randomized functionalities). Specifically, in the worst case scenario, the adversary knows the inputs of all the honest parties. In the next claim we show a lower bound on the probability that all the values that the adversary obtains in a round $i < i^\star$ of Protocol $\mathrm{MPCWithD}_r$ are all equal to a fixed value.

Claim C.2 *Let $d(n)$ and $g(n)$ be the size of the domain and range, respectively, of a randomized functionality $\mathcal{F}$ computed by the protocol $\mathrm{MPCWithD}_r$. Let ϵ be a number such that $\Pr[f_n(x_1, \ldots, x_m) = w_\ell] \geq \epsilon$ for every set of inputs $x_1, \ldots, x_m$ and for each w_ℓ from the range of $f_n(x_1, \ldots, x_m)$. Then, the probability that in a round $i < i^\star$ all the values that the adversary sees are equal to a specific w is at least $(\epsilon/d(n)^m)^{2^t-1}$.*

Furthermore, if $\mathcal{F}$ is deterministic, then, this probability is at least $(1/d(n)^m)^{2^t-1}$.

Proof: We start with the case of a deterministic functionality $\mathcal{F}$. Recall that $x_1, \ldots, x_m$ are the inputs used by the dealer to obtain $w = f_n(x_1, \ldots, x_m)$ and $\sigma_J^{i^\star} = w$ for each $J \subseteq [m]$ s.t. $m - t \leq |J| \leq t$. Let J be such that the adversary obtains σ_J^i in round $i < i^\star$. Recall that $\widehat{x}_1, \ldots, \widehat{x}_m$ are the inputs used by the dealer to obtain σ_J^i, that is, $\sigma_J^i = f_n(\widehat{x}_1, \ldots, \widehat{x}_m)$, where $\widehat{x}_j = x_j$ for each $j \in J$ and $\widehat{x}_j$ is selected uniformly at random from $\widehat{x}_j$ for every $j \notin J$. We bound the probability that $\sigma_J^i = w$ by the probability that $\widehat{x}_j = x_j$ for all $j \notin J$. The probability that $\widehat{x}_j = x_j$ is $1/d$. Therefore, the probability that both sets are the same is $(1/d)^{m-|J|} > (1/d)^m$.

In each round of the protocol, $\mathcal{A}$ obtains the value σ_J^i for each subset Q_J s.t. $J \subseteq [m]$ and $m - t \leq |J| \leq t$, therefore, $\mathcal{A}$ obtains less than 2^t values. For each such two values σ_J^i and $\sigma_{J'}^i$ obtained by $\mathcal{A}$ in round $i < i^\star$, the sets of inputs $\{\widehat{x}_j : j \notin J\}$ and $\{\widehat{x}_j : j \notin J'\}$ are totally independent. Therefore, the probability that all the values that the adversary sees in round $i < i^\star$ are equal to $w = f_n(x_1, \ldots, x_m)$ is at least $(1/d^m)^{2^t-1}$.

For randomized functionality $\mathcal{F}$, we think of the evaluation of $f_n(\widehat{x}_1, \ldots, \widehat{x}_m)$ as two steps: first $\widehat{x}_j$ is randomly chosen from X_n for every $j \notin J$ and then the randomized functionality is evaluated. Therefore, as $\mathcal{A}$ obtains less than 2^t values in each round $i < i^\star$, that the probability that all the values that the adversary sees in each round $i < i^\star$ are equal to the specific w is at least $(1/d^m)^{2^t-1} \cdot \epsilon^{2^t-1}$. □

In the next lemma, we prove the correctness of the simulation by using the previous two lemmas.

Lemma C.3 *Let $\mathcal{F}$ be a (possibly randomized) functionality, $\mathcal{A}$ be a non-uniform polynomial-time adversary corrupting $t < 2m/3$ parties in an execution of Protocol* $\mathrm{MPCWithD}$*, and $\mathcal{S}_T$ be the simulator described in Appendix C.1 (where $\mathcal{S}_T$ controls the same parties as $\mathcal{A}$). Then, for every $n \in \mathbb{N}$, for every $\vec{y} \in (X_n)^m$, and for every* $\mathrm{aux} \in \{0,1\}^*$

$$\mathrm{SD}\left(\mathrm{REAL}_{\mathrm{MPCWithD}_r,\mathcal{A}(\mathrm{aux})}(\vec{y},1^n),\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}_T(\mathrm{aux})}(\vec{y},1^n)\right) \leq 2g(n)d(n)^m/\left(r(n)\right)^{2^t},$$

where $d(n)$ and $g(n)$ are the sizes of the range and the domain of $\mathcal{F}$, respectively, and $r(n)$ be the number of rounds in the protocol.

Furthermore, if $\mathcal{F}$ is deterministic, then, the statistical distance between these two random variables is at most $(d(n)^m)^{2^t}/r(n)$.

Proof: Our goal here is to show that the statistical distance between the above two random variables is at most as described in lemma. The flow of our proof is as follows. We first bound the statistical distance between the two random variables by the probability that the adversary $\mathcal{A}$ guesses the special round $i^\star$. We do this by showing that, conditioned on the event that the adversary fails to guess round $i^\star$, the two random variables are identically distributed. Then, we bound the probability of guessing $i^\star$ in time using Claim C.1 and Claim C.2.

Observe that, in the simulation, $\mathcal{S}_T$ follows the same instructions as the trusted party T in Protocol $\mathrm{MPCWithD}_r$, except for two changes. First, $\mathcal{S}_T$ does not compute the output w_S, but rather gets w_S externally from $T_\mathcal{F}$. The simulator obtains this value either in the premature termination phase (if $i < i^\star$) or in the peeking stage when $i = i^\star$. The second difference is that in the case of a premature termination, $\mathcal{S}_T$ will always use w_S as its message to the corrupt parties, while T will use the value from round $i^\star - 1$ of the appropriate subset Q_J as its message.

We analyze the probabilities of (v, c) in the two random variables according to weather the premature termination occurred before, during, or after the special round $i^\star$.

Premature termination before round $i^\star$. We argue that in this case, both in the real protocol and in the simulation, the view of $\mathcal{A}$ is identically distributed in the two worlds. $\mathcal{S}_T$ follows the same random process in interacting with $\mathcal{A}$ (before sending the last message in the premature termination) as does T in the real-world execution. The view of the adversary consists of values which are outputs of evaluations of the function f_n on the same input distributions. The adversary does not learn anything about the inputs of the honest parties, hence, its decision to abort does not depend on any new information it obtains during the interaction rounds so far. In addition, in both worlds, the output of the honest parties is the evaluation of the function f_n on the same set of inputs for the active parties and uniformly selected random inputs for the aborted parties.

Premature termination after round $i^\star$ or never occurs. Here v must contain $\sigma_J^{i^\star}$ for some J, which, in the real-world execution, is equal to the output value of all sets for any round $i > i^\star$ (recall that the output value of the honest parties will be determined by one such value), and in the simulation it equals w_S. Thus, in both scenarios, v must be consistent with $i^\star$ and with c, hence, v completely determines C. Again, since $\mathcal{S}_T$ follows the same random process in interacting with $\mathcal{A}$ as does T in the real-world execution the probabilities are the same.

Premature termination in round $i^\star$. This is the interesting case, which causes the statistical distance. In the real world, the output of the honest parties is $\sigma_J^{i^\star-1}$ for some J, while in the ideal world their output is $w_S \leftarrow f_n(x_1, \ldots, x_m)$. In the first case the output is independent of the adversary's view, while in the

second case, the view determines the output. Thus, in this case the probabilities of the views are different. However, we will show that the event of premature termination in round $i^\star$ happens with small probability.

Since the probabilities of (v, c) in the first two cases are equal, the statistical distance between the two random variables is bounded by the probability of the adversary guessing $i^\star$ correctly (before the abort phase of round $i^\star$). That is,

$$\mathrm{SD}\left(\mathrm{IDEAL}, \mathrm{REAL}\right) \leq \Pr[\text{Premature termination in round } i^\star]. \tag{1}$$

We next use Claim C.1 and Claim C.2 to bound the probability that the adversary guesses $i^\star$. However, there might be values such that $\Pr[w = f_n(x_1, \ldots, x_m)]$ is small. Therefore, we consider two events of guessing $i^\star$, where p_0 is a parameter specified below. We call an output values w *heavy* if $\Pr[w = f_n(x_1, \ldots, x_m)] > 1/(p_0 \cdot g)$, otherwise, we call w *light*.

Case 1: The adversary guesses $i^\star$ with some light w. Since there are at most g possible values of $f_n(x_1, \ldots, x_m)$, the probability of this event, by the union bound, is at most $1/p_0$.

Case 2: The adversary guesses $i^\star$ with some heavy w. Thus, by Claim C.2 where $\epsilon = p_0 \cdot g$, the probability of $w = \sigma_J^i$ for all values that the adversary sees in round $i < i^\star$ is at least $(1/d^m \cdot p_0 \cdot g)^{2^t-1}$. By Claim C.1, the probability that the adversary guesses $i^\star$ conditioned on the w being heavy is at most $(d^m \cdot p_0 \cdot g)^{2^t-1}/r$.

We take $p_0 = r^{2^{-t}}/(g \cdot d^m)$; the total probability that the adversary guesses $i^\star$ in the two cases is at most

$$\frac{(d^m \cdot p_0 \cdot g)^{2^t-1}}{r} + \frac{1}{p_0} \leq 2 \cdot \frac{g \cdot d^m}{r^{2^{-t}}}.$$

Therefore, by Equation (1), the statistical distance between the two random variables in the randomized case is as claimed in the lemma.

The case that $\mathcal{F}$ is deterministic is simpler. By combining Claim C.1 and Claim C.2 we get that the probability that $\mathcal{A}$ guesses $i^\star$ is at most $(r/d(n)^m)^{2^t-1}$. By applying Equation (1), we get the bound on statistical distance between the two random variables for the deterministic case as claimed in the lemma. □

C.3 The Simulator for the Protocol with the Dealer for Polynomial Range

Lemma C.4 *Let $\mathcal{F}$ be a (possibly randomized) functionality. For every non-uniform polynomial-time adversary $\mathcal{A}$ corrupting $t < 2m/3$ parties in an execution of Protocol* MPCWithDForRange*, there exists a simulator $\mathcal{S}_T$ in the ideal model, that simulates the execution of $\mathcal{A}$ (where $\mathcal{S}_T$ controls the same parties as $\mathcal{A}$). That is, for every $n \in \mathbb{N}$, for every $\vec{y} \in (X_n)^m$, and for every* $\mathrm{aux} \in \{0,1\}^*$

$$\mathrm{SD}\left(\mathrm{REAL}_{\mathrm{MPCWithD}_r, \mathcal{A}(\mathrm{aux})}(\vec{y}, 1^n), \mathrm{IDEAL}_{\mathcal{F}, \mathcal{S}_T(\mathrm{aux})}(\vec{y}, 1^n)\right) < \frac{(2p(n) \cdot g(n))^{2^t}}{r(n)} + \frac{1}{2p(n)},$$

where $g(n)$ is the size of the range of $\mathcal{F}$, with probability $1/(2p(n))$ each value σ_J^i in round $i < i^\star$ is selected uniformly at random from the range, and $r(n)$ be the number of rounds in the protocol.

Proof: The simulators and their proofs for Protocol MPCWithDForRange and Protocol MPCWithD are similar; we only present (informally) the differences between the two simulators and the two proofs.

The modified simulator. Recall that the protocols $\mathrm{MPCWithD}$ and $\mathrm{MPCWithDForRange}$ are different only in Step (3) of the share generation step. In $\mathrm{MPCWithDForRange}$, each value σ_J^i prior to round $i^\star$ is chosen with probability $1/(2p)$ as a random value from the range of f_n and with probability $1 - 1/(2p)$ it is chosen just like in Figure 1. There are two modifications to the simulator. The first modification in the simulator is in Step (6) in the simulation of the preprocessing phase, i.e., in the computation of σ_J^i for $i < i^\star$. The step that replaces Step (6) appears below.

- For each $i \in \{1, \ldots, i^\star - 1\}$ and for each $J \subseteq B \setminus D_0$ s.t. $m - t \leq |J| \leq t$ do
  1. with probability $1/(2p)$, select uniformly at random $z_J^i \in Z_n$ and set $\sigma_J^i = z_J^i$.
  2. with the remaining probability $1 - 1/(2p)$,
     (a) For each $j \in [m]$, if $j \in J$, then $\mathcal{S}_T$ sets $\widehat{x}_j = x_j$, else, $\mathcal{S}_T$ selects uniformly at random $\widehat{x}_j \in X_n$.
     (b) $\mathcal{S}_T$ sets $\sigma_J^i \leftarrow f_n(\widehat{x}_1, \ldots, \widehat{x}_m)$.

The second modification is less obvious. Recall that both random variables appearing in the lemma contain the output of the honest parties. In the ideal world, the honest parties always output f_n applied to their inputs. In the real world, in a premature termination in round $i < i^\star$, with probability $1/(2p)$, the honest parties output a random value from the range of f_n. It is hard to simulate the output of the honest parties in first case.[4] We simply modify the simulator such that with probability $1/(2p)$ the simulator returns $\perp$, i.e., it announces that the simulation has failed. The new premature termination step appears below.

Simulating the premature termination step:

- If the premature termination step occurred in round $i < i^\star$,
  - With probability $1/(2p)$, for each $j \in B \setminus D_0$ send "$\texttt{abort}_j$" to the trusted party computing $\mathcal{F}$ and return $\perp$.
  - With the remaining probability $1 - 1/(2p)$, execute the original simulation of the premature termination step (appearing in Appendix C.1).
- Else ($i \geq i^\star$), execute the original simulation of the premature termination step (appearing in Appendix C.1).

The modified proof. The proof to the simulator for $\mathrm{MPCWithDForRange}$ remains basically the same, except for two changes. We first modify Claim C.2 below and prove a slightly different claim, which changes the probability of the adversary guessing $i^\star$.

Claim C.5 *Let $g(n)$ be the size of the range of the (possibly randomized) functionality $\mathcal{F}$ computed by the protocol $\mathrm{MPCWithDForRange}_r$ and $w \in Z_n$. Then, the probability that in a round $i < i^\star$ all the values that the adversary sees are equal to w is at least $(1/2p(n) \cdot g(n))^{2^t}$.*

Proof: According to the protocol, there are two different ways to produce each value σ_J^i in round $i < i^\star$: (1) Compute f_n on a set of inputs and a set of uniformly selected values from the domain of the functionality, and (2) Set σ_J^i as a uniformly selected value from the range of the functionality. We ignore the first case. In the second option, with probability $1/2p$, the value σ_J^i is uniformly selected from the range. Hence, the probability that σ_J^i is equal to a specific value is at least $1/(2p \cdot g)$.

[4] For example, there might not be possible inputs of the corrupt parties causing the honest parties to output such output.

It was explained in the proof of Claim C.2 that in each round of the protocol, $\mathcal{A}$ obtains less than 2^t values. Therefore, we conclude that he probability that all the values that $\mathcal{A}$ obtains in round $i < i^\star$ are all equal to w is at least $(1/(2p \cdot g))^{2^t}$. □

By applying the Claim C.1 we conclude that the probability of the adversary guessing $i^\star$ correctly in Protocol $\mathrm{MPCWithDForRange}_r$ is at most $(2p \cdot g)^{2^t}/r$. In case of a premature termination in round $i < i^\star$, with probability $1 - 1/(2p)$ in both the ideal world and real world, the value that the honest parties output is the evaluation of f_n on the inputs of the active parties and random inputs for the parties that aborted. However, with probability $1/(2p)$, if premature termination occurs prior to round $i^\star$, the output of the honest parties Protocol $\mathrm{MPCWithDForRange}_r$ is a random value from the range of f_n; the simulator fails to simulate the execution in this case and outputs $\perp$. Thus,

$$\begin{aligned}
&\mathrm{SD}\left(\mathrm{IDEAL}, \mathrm{REAL}\right) \\
&\quad \leq \Pr[\text{Premature termination in round } i^\star] + (1/2p) \cdot \Pr[\text{Premature termination before round } i^\star] \\
&\quad \leq (2p \cdot g)^{2^t}/r + (1/2p).
\end{aligned}$$

Therefore, the statistical distance is as claimed. □

D Proof of Security for the Protocols without the Dealer

D.1 The Simulator for Protocol MPC_r

We next prove that Protocol MPC_r is a secure real-world implementation of the (ideal) functionality of Protocol $\mathrm{MPCWithD}_r$. By Lemma C.3, when $r(n)$ is sufficiently large, Protocol $\mathrm{MPCWithD}_r$ is a $1/p$-secure protocol for $\mathcal{F}$. Thus, together we get that Protocol MPC_r is a $1/p$-secure protocol for $\mathcal{F}$. according to the definition appears in Appendix A. We analyze Protocol MPC_r in a hybrid model where there are 3 ideal functionalities:

Functionality $\mathbf{MultiShareGenWithAbort_r}$. This functionality is an (ideal) execution of Functionality $\mathrm{MultiShareGen}_r$ in the secure-with-abort and cheat-detection model. That is, the functionality gets a set of inputs. If the adversary sends "$\mathtt{abort}_j$" for some corrupt party p_j, then this message is sent to the honest parties and the execution terminates. Otherwise, Functionality $\mathrm{MultiShareGen}_r$ is executed. Then, the adversary gets the outputs of the corrupt parties. Next, the adversary decides whether to halt or to continue: If the adversary decides to continue, it sends a "$\mathtt{proceed}$" message and the honest parties are given their outputs. Otherwise, the adversary sends "$\mathtt{abort}_j$" for some corrupt party p_j, and this message is sent to the honest parties.

Functionality FairMPC. This functionality computes the value $f_n(x_1, \ldots, x_m)$. That is, the functionality gets a set of inputs. If a party p_j sends "$\mathtt{abort}_j$" message then x_j selected from X_n with uniform distribution, computes an output of the randomized functionality f_n for them, and gives it to all parties. When this functionality is executed, an honest majority is guaranteed, hence, the functionality can be implemented with full security (e.g., with fairness).

Functionality Reconstruction. This functionality is described in Figure 4; this functionality is used in the premature termination step in Protocol MPC_r for reconstructing the output value from the shares of the previous round. When this functionality is executed, an honest majority is guaranteed, hence, the functionality can be implemented with full security (e.g., with fairness).

We consider an adversary $\mathcal{A}$ in the hybrid model described above, corrupting $t < 2m/3$ of the parties that engage in Protocol MPC_r. We next describe a simulator $\mathcal{S}$ interacting with the honest parties in the ideal-world via a trusted party T_{MPCWithD} executing Functionality $\mathrm{MPCWithD}_r$. The simulator $\mathcal{S}$ runs the adversary $\mathcal{A}$ internally with black-box access. Simulating $\mathcal{A}$ in an execution of the protocol, $\mathcal{S}$ corrupts the same subset of parties as does $\mathcal{A}$. Denote by $B = \{i_1, \ldots, i_t\}$ the set of indices of corrupt party. At the end of the computation it outputs a possible view of the adversary $\mathcal{A}$. To start the simulation, $\mathcal{S}$ invokes $\mathcal{A}$ on the set of inputs $\{y_j : j \in B\}$, the security parameter 1^n, and the auxiliary input aux.

Simulating the preliminary phase:

1. $D_0 = \emptyset$.
2. The simulator $\mathcal{S}$ receives a set of inputs $\{x_j : j \in B \setminus D_0\}$ that $\mathcal{A}$ submits to Functionality $\mathrm{MultiShareGenWithAbort}_r$.
   If a party p_j for $j \in B \setminus D_0$ does not submit an input, i.e., sends an "$\texttt{abort}_j$" message, then,
   (a) $\mathcal{S}$ sends "$\texttt{abort}_j$" to the trusted party T_{MPCWithD}.
   (b) $\mathcal{S}$ updates $D_0 = D_0 \cup \{j\}$.
   (c) If $|D_0| < m - t$, then Step (2) is repeated.
   (d) Otherwise ($|D_0| \geq m - t$), simulate premature termination with $i = 1$.
3. $\mathcal{S}$ prepares outputs for the corrupted parties for Functionality $\mathrm{MultiShareGenWithAbort}_r$: The simulator $\mathcal{S}$ sets $\sigma_J^i = 0$ for every $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$ and for all $i \in \{1, \ldots, r\}$. Then, $\mathcal{S}$ follows Step (1) and Steps 5–8 in the computation of Functionality $\mathrm{MultiShareGen}_r$ (skipping the Steps 2–4) to obtain shares for the parties.[5]
4. For each party p_j s.t. $j \in B \setminus D_0$, the simulator $\mathcal{S}$ sends to $\mathcal{A}$:
   - The verification key K_{ver}.
   - The masking shares $\mathrm{mask}_j(R_j^{i,J})$ for each $i \in \{1, \ldots, r\}$ and for every $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$ and $j \in J$.
   - The messages $M_{j,1}, \ldots, M_{j,r}$.
5. If $\mathcal{A}$ sends an "$\texttt{abort}_j$" for some party p_j s.t. $j \in B \setminus D_0$ to $\mathcal{S}$, then,
   (a) $\mathcal{S}$ sends "$\texttt{abort}_j$" to the trusted party T_{MPCWithD}.
   (b) $\mathcal{S}$ updates $D_0 = D_0 \cup \{j\}$.
   (c) If $|D_0| < m - t$, then Steps 2–5 are repeated.
   (d) Otherwise ($|D_0| \geq m - t$), go to simulating premature termination with $i = 1$.

   Otherwise ($\mathcal{A}$ sends a "$\texttt{continue}$" message to $\mathcal{S}$),
   (a) The simulator $\mathcal{S}$ denotes $D = D_0$.
   (b) The simulator sends x_j to T_{MPCWithD} for every $j \in B \setminus D_0$ (and gets as response a "$\texttt{proceed}$" message).

Simulating interaction rounds:

Let $\mathcal{J}$ be the collection of subsets $J \subseteq B \setminus D_0$ s.t. $m - t \leq |J| \leq t$. I.e., $\mathcal{J}$ is the collection of sets of indices of active corrupt parties after the simulation of the executions of $\mathrm{MultiShareGenWithAbort}_r$
To simulate round i for $i = 1, \ldots, r$, the simulator $\mathcal{S}$ proceeds as follows:

[5] These shares are temporary and will later be open to the actual values obtained from T_{MPCWithD} during the interaction rounds using the properties of Shamir's secret-sharing scheme.

1. $\mathcal{S}$ gets from the trusted party T_{MPCWithD} the values that the corrupted parties see. That is, $\mathcal{S}$ gets a bit τ_J^i for each $J \in \mathcal{J}$.[6]
2. The simulator $\mathcal{S}$ selects shares for the inner secret-sharing scheme for corrupted parties: For every $J \in \mathcal{J}$, the simulator $\mathcal{S}$ selects uniformly at random shares of τ_J^i in a $|J|$-out-of-$|J|$ Shamir secret sharing scheme. Denote these shares by $\left\{X_j^{i,J} : p_j \in Q_J\right\}$.

   For each $p_j \in Q_J$, let $Y_j^{i,J} \leftarrow (X_j^{i,J}, i, J, j, \mathrm{Sign}((X_j^{i,J}, i, J, j), K_{\mathrm{sign}}))$.
3. The simulator $\mathcal{S}$ selects complementary shares for all honest parties: For every $J \in \mathcal{J}$ and for each $j \in B \setminus D_0$,
   (a) S calculates $\alpha_j = \mathrm{mask}_j(R_j^{i,J}) \oplus Y_j^{i,J}$.
   (b) S selects uniformly at random $m - t$ shares of α_j uniformly at random over all possible selections of $m - t$ shares that are shares of α_j together with the $|B \setminus D_0| - 1$ shares

   $$\left\{\mathrm{comp}_q(R_j^{i,J}) : q \in B \setminus (D_0 \cup \{j\})\right\}$$

   produced in Step (3) in the simulation of the preliminary phase.
   (This is possible according to the property of Shamir's scheme)
   Denote by $\mathrm{comp}_q(Y_j^{i,J})$ the complementary share that $\mathcal{S}$ selects for the honest party p_q for a party p_j s.t. $j \in (B \setminus D_0) \cap J$, where $J \in \mathcal{J}$.
4. For party p_j and a subset $J \notin \mathcal{J}$, let $\mathrm{comp}_q(R_j^{i,J})$ be the complementary share which was produced in Step (3) in the simulation of the preliminary phase, i.e., $\mathrm{comp}_q(R_j^{i,J})$.
5. Construct signed messages $m'_{q,i}$ for each honest party p_q in round i by concatenating:
   (a) q.
   (b) The round number i.
   (c) The complement shares which were described in Step (4) above.
   (d) The complement shares $\mathrm{comp}_q(Y_j^{i,J})$ for all $J \in \mathcal{J}$ and for all $j \in J$ produced in Step (3) for p_q.

   Then, $\mathcal{S}$ signs $m'_{q,i}$, i.e., $\mathcal{S}$ computes $M'_{q,i} \leftarrow (m'_{q,i}, \mathrm{Sign}(m'_{q,i}, K_{\mathrm{sign}}))$.
6. The simulator $\mathcal{S}$ sends all the message $M'_{q,i}$ on behalf of each honest party p_q to $\mathcal{A}$.
7. For every $j \in B \setminus D_0$ s.t. $\mathcal{A}$ sends an invalid or no message on behalf of p_j, the simulator $\mathcal{S}$ sends "$\mathtt{abort}_j$" to T_{MPCWithD}:
   (a) $D = D \cup \{j\}$.
   (b) If $|D| \geq m - t$ go to premature termination step.
   (c) Otherwise, the simulator $\mathcal{S}$ proceeds to the next round.

Simulating the premature termination step:

- If $i = 1$, then S simulates $\mathcal{A}$'s interaction with Functionality $\mathrm{FairMPC}$ as follows:
  1. S receives from $\mathcal{A}$ the inputs of the active corrupt parties.
  2. For every $j \in B \setminus D$: If p_j does not send an input, then $\mathcal{S}$ sends "$\mathtt{abort}_j$" to T_{MPCWithD} else, $\mathcal{S}$ sends p_j's input to T_{MPCWithD}.

[6] In Steps 2–5, the simulator $\mathcal{S}$ constructs the messages of the honest parties in order to allow the corrupted parties in each $J \in \mathcal{J}$ to reconstruct τ_J^i.

- If $i > 1$, then S simulates $\mathcal{A}$'s interaction with Functionality $\mathrm{Reconstruction}$ as follows:
  1. S receives from $\mathcal{A}$ the inputs of the active corrupt parties, i.e., p_j s.t. $j \in B \setminus D$.
  2. If an active corrupt party p_j, does not send an input, or its input is not appropriately signed or malformed, then $\mathcal{S}$ sends "$\texttt{abort}_j$" to T_{MPCWithD}.
- $\mathcal{S}$ gets from T_{MPCWithD} a value σ and sends it to $\mathcal{A}$.
- The simulator $\mathcal{S}$ outputs the sequence of messages exchanged between $\mathcal{S}$ and the adversary $\mathcal{A}$ and halts.

Simulating normal termination at the end of round r:

1. The simulator gets w from the trusted party T_{MPCWithD}.
2. $\mathcal{S}$ constructs all the singed shares of the inner secret-sharing scheme for each $J \subseteq [m] \setminus D_0$ s.t. $m - t \leq |J| \leq t$ and for each honest party $p_j \in Q_J$ as follows.
   For each $J \notin \mathcal{J}$, the simulator S selects uniformly at random $|J \setminus B|$ shares of w uniformly at random over all possible selections of $|J \setminus B|$ shares that together with the $|J \cap B|$ given shares $\left\{R_j^{i,J} : j \in B\right\}$ (produced in Step (2) in the simulation of the preliminary phase) are a sharing of w in a $|J|$-out-of-$|J|$ secret sharing scheme.
   (This is possible according to the property of Shamir's scheme)
   Denote these shares by $\left\{X_j^{r,J}\right\}$.

   For each share $X_j^{r,J}$, the simulator concatenates the corresponding identifying details, and signs them to obtain: $Y_j^{r,J} \leftarrow (X_j^{r,J}, r, J, j, \mathrm{Sign}((X_j^{r,J}, r, J, j), K_{\mathrm{sign}}))$.
3. For each honest party p_j, the simulator $\mathcal{S}$ sends to $\mathcal{A}$ the shares $Y_j^{r,J}$ for all subsets J, such that $p_j \in Q_J$.
4. The simulator $\mathcal{S}$ outputs the sequence of messages exchanged between $\mathcal{S}$ and the adversary $\mathcal{A}$ and halts.

D.2 Proving the Correctness of Protocol MPC_r and Protocol $\mathrm{MPCForRange}_r$

It can be proved that Protocol MPC_r is a secure implementation of the (ideal) functionality of the dealer's in Protocol $\mathrm{MPCWithD}_r$. That is,

Lemma D.1 *Let $t < 2m/3$. If enhanced trap-door permutations exist, then Protocol MPC_r presented in Section 3.2, is a computationally-secure implementation (with full security) of the dealer functionality in Protocol $\mathrm{MPCWithD}_r$.*

In [3], a similar framework to the one used in this paper is used: first a protocol with a dealer for the coin-tossing problem is presented and, then, a real-world protocol that is a computationally-secure implementation (with full security) of the dealer functionality is described. In [3], a simulator for this protocol is given. This simulator is similar to the simulator described in Appendix D.1, than a full proof for the simulator is provided. As the proof is very similar to the proof of our simulator, we omit the proof.

To conclude the proof, as $\mathrm{MPCWithD}_r$ is a $1/p$-secure implementation of $\mathcal{F}$ and MPC_r is a secure implementation of the (ideal) functionality of the dealer in Protocol $\mathrm{MPCWithD}_r$, by the composition theorem of Canetti [8] we conclude that MPC_r $1/p$-secure implementation of $\mathcal{F}$. That is, Theorem 1 is proved.

Next, we claim that $\mathrm{MPCForRange}_r$ is a secure implementation of the (ideal) functionality of the dealer in Protocol $\mathrm{MPCWithDForRange}_r$. That is,

Lemma D.2 *Let $t < 2m/3$. If enhanced trap-door permutations exist, then Protocol $\mathrm{MPCForRange}_r$ described in Section 3.3, is a computationally-secure implementation (with full security) of the dealer functionality in Protocol $\mathrm{MPCWithDForRange}_r$.*

Proof: Recall that the only difference between Protocol MPC_r and Protocol $\mathrm{MPCForRange}_r$ is in the way that the values that the parties see prior round $i^\star$ are produced, i.e., the difference is in Functionality $\mathrm{MultiShareGen}_r$. Specifically, in Section 3.3 we presented a modification in Step (3) in Functionality $\mathrm{MultiShareGen}_r$ in order to get Protocol MPC_r from Protocol $\mathrm{MPCForRange}$. Now, observe that the simulator presented above does not refer to Step (3) of Functionality $\mathrm{MultiShareGen}_r$ in any step. Therefore, the simulator presented in Appendix D.1 for Protocol MPC_r is also a simulator for Protocol $\mathrm{MPCForRange}_r$. □

Claim C.5 and Lemma D.2 imply Theorem 2.