1/p-Secure Multiparty Computation without Honest Majority and the Best of Both Worlds

Amos Beimel[1], Yehuda Lindell[2], Eran Omri[2], and Ilan Orlov[1]

[1] Dept. of Computer Science, Ben Gurion University*
[2] Dept. of Computer Science, Bar Ilan University**

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 that are resilient against *any* number of corrupt parties provided that the number of parties is constant and the size of the range of the functionality is at most polynomial (in the security parameter n). If less than 2/3 of the parties are corrupt, the size of the domain is constant, and the functionality is deterministic, then our protocols are efficient even when the number of parties is $\log\log n$. 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. Thus, our feasibility results for 1/p-secure computation are essentially tight.

We further motivate our results by constructing protocols with stronger guarantees: If in the execution of the protocol there is a majority of honest parties, then our protocols provide full security. However, if only a minority of the parties are honest, then our protocols are 1/p-secure. Thus, our protocols provide the best of both worlds, where the 1/p-security is only a fall-back option if there is no honest majority.

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

* Generously supported by ISF grant 938/09 and by the Frankel Center for Computer Science.

** Generously supported by the European Research Council as part of the ERC project LAST, and by ISF grant 781/07.

inputs to a trusted party which, in turn, 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 [13]. Secure computation is fair – all parties get the output. Cleve [7] proved that, in general, fairness is not possible without an honest majority.

To overcome the impossibility of [7], Gordon and Katz [18] 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 efficiently 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 [7] 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 [7, 8, 23, 3], for contract signing/exchanging secrets [5, 22, 10, 4, 9, 6], and for general functionalities [26, 11, 1, 14, 25, 12, 18]. We next describe the papers that are most relevant to our paper. Moran, Naor, and Segev [23] 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 [18] 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, Beimel, Omri, and Orlov [3] construct multiparty protocols for coin tossing that are $O(1/r)$-secure provided that the fraction of corrupt parties is slightly larger than half. In particular, their 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. [15] showed that complete fairness is possible in the two party case for some functions. Gordon and Katz [17] showed similar results for the multiparty case. The characterization of the functions that can be computed with full fairness without honest majority is open. Gordon et al. [16] studied completeness for fair computations. Specifically, they showed a specific function that is complete for fair two-party computation; this function is also complete for $1/p$-secure two-party computation.

Ishai et al. [19] considered "best of two worlds" protocols. Such protocols should provide full security with an honest majority and some (weaker) security if there is only a minority of honest parties. They give positive and negative results for the existence of such protocols. We discuss some of their results below.

1.1 Our Results

We study $1/p$-secure protocols in the multiparty setting. We construct protocols for general functionalities that are $1/p$-secure against *any* number of corrupt parties provided that the number of parties is constant. Our protocols require that the size of the range of the (possibly randomized) functionality is at most polynomial in the security parameter. That is, we show the following feasibility result.

Theorem (Informal). *Let $\mathcal{F}$ be a (possibly randomized) functionality with a constant number of parties whose size of range is at most polynomial in the security parameter n. Then, for every polynomial $p(n)$ there is a $1/p(n)$-secure protocol for $\mathcal{F}$ tolerating any number of corrupt parties.*

Our results are the first general feasibility results for $1/p$-secure protocols in the multi-party setting, e.g., even for the case that there are 3 parties and two of them might be corrupt. We provide two additional protocols that are $1/p$-secure assuming that the fraction of corrupt parties is less than 2/3. These two protocols are more efficient than the protocols discussed above. Specifically, one of the protocols is $1/p$-secure even when the number of parties is $\log \log n$ (where n is the security parameter) provided that the functionality is deterministic and the size of the domain of inputs is constant.

The definition of $1/p$-security allows that with probability $1/p$ the outputs of the honest parties will be arbitrary, e.g., for a Boolean function the outputs can be non-Boolean. Some of our protocols are always correct, that is, they always return an output of the functionality with the inputs of the honest parties and some inputs for the corrupt parties. This correctness property is essential for the best of both worlds results described below.

We further motivate our results by constructing protocols with best of both worlds guarantees: If in the execution of the protocol there is a majority of honest parties, then our protocols provide full security. However, if only a minority of parties are honest, then our protocols are $1/p$-secure. The protocols succeed although they do not know in advance if there is an honest majority or not. Specifically, we show that

Theorem (Informal). *Let $\mathcal{F}$ be a functionality with a constant number of parties whose size of domain and range is at most polynomial in the security parameter n. Then, for every polynomial $p(n)$ there is a protocol for $\mathcal{F}$ tolerating any number of corrupt parties such that*

- *If there is an honest majority, then the protocol is fully secure.*
- *If there is no honest majority, then the protocol is $1/p(n)$-secure.*

Thus, the $1/p$-security guarantee can be considered as a fall-back option if there is no honest majority. Our protocols provide the best of both worlds, the world of honest majority where the known protocols (e.g., [13]) provide full security if there is an honest majority and provide no security guarantees if no such majority exists and the world of secure computation without honest majority.

In the latter world the security is either security-with-abort or $1/p$-security. These types of security are incomparable. Ishai et al. [19] proved that there is no general protocol which provides full security when there is an honest majority and security-with-abort without an honest majority. Thus, our protocols provide the best possible combination of both worlds.

Katz [21] presented a protocol, for any functionality $\mathcal{F}$, with full security when there is an honest majority, as well as $1/p$-security *with abort* for any number of corrupt parties. This result assumes a non-rushing adversary. In contrast, our protocols achieve a stronger security with a minority of honest parties and can handle the more realistic case of a rushing adversary. However, our protocols only work with a constant number of parties and a polynomial size domain.

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 that 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 results is rather simple and follows from an impossibility result of [18]. Nevertheless, they show that our general feasibility results are almost tight.

Our impossibility results should be contrasted with the coin-tossing protocol of [3] 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.

The above mentioned impossibility results do not rule out that the best of two worlds results of Katz [21] can be strengthened by removing the restriction that the adversary is non-rushing. We show that this is impossible, that is, in general, when the number of parties is super-constant and the size of the domain is polynomial, there is no protocol that is fully secure with an honest majority and $1/p$-secure-with-abort without such a majority.

The ideas behind our protocols. Our protocols use ideas from the protocols of Gordon and Katz [18] and Beimel et al. [3], both of which generalize the protocol of Moran, Naor, and Segev [23]. In addition, our protocols introduce new ideas that are required to overcome challenges that did not occur in previous works, e.g., dealing with inputs (in contrast to the scenario of [3]) and dealing with a dishonest majority even after parties abort (in contrast to the scenario of [18]). In particular, in order to achieve resilience against any number of corrupt parties we introduce new techniques for hiding the round in which parties learn the output of an execution. 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; starting 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. Similar to the protocols of [23, 18, 3], 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.

In our protocols that are $1/p$-secure against a fraction of $2/3$ corrupt parties (which are described in Section 4), if in some round many (corrupt) parties have aborted and 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. The mechanism to secret share the values in this protocols is similar to [3], however, there are important differences in this sharing, as the sharing mechanism of [3] is not appropriate for $1/p$-secure computations of functionalities which depend on inputs. The fact that the protocol proceeds until there is an honest majority imposes some restrictions that imply that the protocol can tolerate only a fraction of $2/3$ corrupt parties.

Our protocols that are $1/p$-secure against any number of corrupt parties (which are described in Section 5) take a different route. To describe the ideas of the protocol, we consider only the three-party case, where at most two parties are corrupt. In the protocol if one party aborts, then the remaining two parties execute a two-party protocol for the functionality. Again, this protocol proceeds in rounds, where in each round each party gets a value. If the party in the three-party protocol aborts after round $i^\star$, then all these values are the "correct" output of the functionality. To hide $i^\star$, also prior to $i^\star$, with some probability all these values must be equal. With the remaining probability, a new $i^\star$ is chosen with uniform distribution for the two-party protocol. In other words, in the two-party protocol prior to the original $i^\star$, with some probability, we chose a "fake" value of 1 for the new $i^\star$ of the two-party protocol.

Open Problems. In our impossibility results the size of the range is super-polynomial (in the security parameter). However, in all our protocols the size of the range is 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 our protocols, the number of rounds is double-exponential in the number of parties. Our impossibility results do not rule out that this double-exponential dependency can be improved.

The protocols of [18] 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. Some of our protocols are provably not private (that is, the adversary can learn extra information). However, for other protocols, we do not know whether they are private. It is open if there are general multiparty $1/p$-secure protocols that are also private.

2 Background and the Model of Computation

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).

In this work we consider a corrupt, static, computationally-bounded (i.e., non-uniform probabilistic polynomial-time) adversary that controls 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 states of the corrupted parties and fully controls the messages that they broadcast 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 sees the messages broadcast by the honest parties before broadcasting the messages of the corrupted parties for this round (thus, the broadcast messages of the corrupted parties can depend on the messages of the honest parties in the same round).

In this work we consider $1/p$-secure computation. Roughly speaking, we say that a protocol Π is $1/p$-secure if for every adversary $\mathcal{A}$ attacking Π in the real-world there is a simulator $\mathcal{S}$ running in the ideal-world, such that the global output of the real-world and the ideal-world executions cannot be distinguished with probability greater than $1/p$. The formal definitions of $1/p$-security and security with abort and cheat detection, which is a tool used in this paper, will be given in the full version of the paper.

3 Feasibility Results for $1/p$-Secure Multiparty Computation

In this section we state our main feasibility results. Our main result asserts that any functionality with a polynomial size range for a constant number of parties can be $1/p$-securely computed in polynomial time tolerating any number of corrupt (malicious) parties. We next formally state this result.

Theorem 1. *Let $\mathcal{F}$ be an m-party (possibly randomized) functionality. If enhanced trap-door permutations exist, and if m is constant and the size of the range $g(n)$ is bounded by a polynomial in the security parameter n, then for any polynomial $p(n)$ there is an $r(n)$-round $1/p(n)$-secure protocol computing $\mathcal{F}$ tolerating up to $m-1$ corrupt parties, where $r(n) = \Big(p(n) \cdot g(n)\Big)^{2^{O(m)}}$.*

The protocol that implies Theorem 1 for general m will appear in the full version of this paper. In this extended abstract we present, in Section 5, the 3-party version of this protocol tolerating up to 2 corrupt parties. In addition, for functionalities where the domain size is also bounded by a polynomial, we

will present in the full version of this paper a protocol with somewhat stronger security properties. Using these stronger security, we can transform it into a protocol of the best of both worlds type (see Section 6.1 for details).

We give substantially better protocols secure against an adversary that may corrupt strictly less than two-thirds of the parties. Formally, we prove the following theorem.

Theorem 2. *Let $\mathcal{F}$ be an $m(n)$-party (possibly randomized) functionality. Let $t(n)$ be such that $m(n)/2 \leq t(n) < 2m(n)/3$. If enhanced trap-door permutations exist, then for any polynomial $p(n)$ the following hold:*

- *If $m(n)$ is constant (hence, $t = t(n)$ is constant) and the size of the range $g(n)$ is bounded by a polynomial, then there exists an $r(n)$-round $1/p(n)$-secure protocol computing $\mathcal{F}$ tolerating up to t corrupt parties, where $r(n) = (2p(n))^{2^t+1} \cdot g(n)^{2^t}$.*
- *If $\mathcal{F}$ is deterministic and the size of the domain $d(n)$ is bounded by a polynomial, then there exists an $r(n)$-round $1/p(n)$-secure protocol computing $\mathcal{F}$ tolerating up to $t(n)$ corrupt parties, where $r(n) = p(n) \cdot d(n)^{m(n)\cdot 2^{t(n)}}$, provided that $r(n)$ is bounded by a polynomial.*

The protocols that imply the results of Theorem 2 are presented in Section 4. As implied by the second item of Theorem 2, the round complexity of our protocol when $\mathcal{F}$ is deterministic has only a linear dependency on $p(n)$. Specifically, this protocol has polynomially many rounds even when the number of parties is $0.5 \log \log n$ provided that the functionality is deterministic and the size of the domain of inputs is constant.

4 Protocols with Less Than Two-Thirds Corrupt Parties

In this section we describe our protocols that are secure when the adversary corrupts strictly less than two thirds of the parties. 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 $1 - 1/p$.

Following [23, 3], we present the first protocol in two stages. We first describe in Section 4.1 a protocol with a dealer and then in Section 4.2 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 us to prove the protocol in a modular way. In Section 4.3, 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).

4.1 The Protocol for Polynomial-Size Domain with a Dealer

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 4.2, 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 in 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 [21, 15, 23, 18, 3], 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 can cause harm only if it guesses $i^\star$ and this happens with small probability.

In this extended abstract, we only give a verbal description of the protocol. This protocol is designed such that the dealer can be removed from it in Section 4.2. 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 an input, then the dealer sets x_j to be a random value selected uniformly from the input domain X_n. In a preprocessing phase, the dealer T selects uniformly at random a special round $i^\star \in \{1, \ldots, r\}$. The dealer computes $w \leftarrow f_n(x_1, \ldots, x_m)$. Then, for every round $1 \le i \le r$ and every $L \subset \{1, \ldots, m\}$ such that $m - t \le |L| \le t$, the dealer selects an output, denoted σ_L^i, as follows (this output is returned by the parties in $Q_L = \{p_j : j \in L\}$ if the protocol terminates in round $i+1$ and Q_L is the set of the active parties):

Case I: $1 \le i < i^\star$. For every $j \in L$ the dealer sets $\widehat{x}_j = x_j$ and for every $j \notin L$ it chooses $\widehat{x}_j$ independently with uniform distribution from the domain X_n; it computes the output $\sigma_L^i \leftarrow f_n(\widehat{x}_1, \ldots, \widehat{x}_m)$.
Case II: $i^\star \le i \le r$. The dealer sets $\sigma_L^i = w$.

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

The peeking phase. The dealer T sends to the adversary all the values σ_L^i such that all parties in Q_L 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_L^{i-1} to the active parties, where Q_L is the set of the active parties, where parties can also abort during this phase. 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 to the next round.

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

Example 1. As an example, assume that $m = 5$ and $t = 3$. In this case the dealer computes a value σ_L^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.

We next hint why for deterministic functionalities, an 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 [23, 18, 3], 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_L^i and $\sigma_{L'}^i$ that the adversary can obtain such that $\sigma_L^i \neq \sigma_{L'}^i$, 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 [18, Lemma 2], this implies that the protocol is $d^{O(1)}/r$-secure.

4.2 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_L^i of each subset Q_L among the parties of Q_L using a $|L|$-out-of-$|L|$ secret-sharing scheme – called *inner* secret-sharing scheme. As in protocol with the dealer (described in Section 4.1), the adversary is able to obtain information on σ_L^i only if it controls all the parties in Q_L. On the other hand, the honest parties can reconstruct σ_L^{i-1} (without the dealer), where Q_L 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_L. 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_L^i in a $|L|$-out-of-$|L|$ secret-sharing scheme, for every set Q_L such that $j \in L$ and $m - t \leq |L| \leq t$ (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_L^i 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. We call the resulting secret-sharing scheme the *outer* $(t+1)$-out-of-m scheme (since t parties and the holder of the mask are needed to reconstruct the secret).

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.

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 $|L|$-out-of-$|L|$ inner secret-sharing of the values σ_L^i (for the appropriate sets Q_L's to which the party belongs). If a party stops broadcasting messages or broadcasts improperly signs messages, then all 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_L^{i-1}. If the premature termination occurs in the first round, then 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 at least $m-t$ 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. This is done similarly to the preprocessing phase in [3], which in turn use the results of [24, 2]. 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.

Comparison with the multiparty coin-tossing protocol of [3]. Our protocol combines ideas from the protocols of [18, 3]. However, there are some important differences between our protocol and the protocol of [3]. In the coin-tossing protocol of [3], the bits σ_L^i are shared using a threshold scheme where the threshold is smaller than the size of the set Q_L. This means that a proper subset of Q_L containing corrupt parties can reconstruct σ_L^i. In coin-tossing this is not a problem since there are no inputs. However, when computing functionalities with inputs, such σ_L^i might reveal information on the inputs of honest parties in Q_L, and we share σ_L^i with threshold $|Q_L|$. As a result, we use more sets Q_L than in [3] 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 [3] 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 7, this difference is inherent as a protocol for general functionalities with polynomially many parties and $t = m/2$ cannot have a small bias.

4.3 A 1/p-Secure Protocol for Polynomial Range

Using an idea of [18], we modify our protocol so 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_L^i is chosen prior to round $i^\star$: with probability $1/(2p)$ we choose σ_L^i as a random value in the range of f_n and with probability $1 - 1/(2p)$ we choose it as in Case I described in Section 4.1. More formally, in the protocol with the dealer, in the preprocessing phase we replace Case I with the following step:

- For each $i \in \{1, \ldots, i^\star - 1\}$ and for each $L \subseteq [m]$ s.t. $m - t \leq |L| \leq t$,
  - with probability $1/(2p)$, the dealer selects uniformly at random $z_L^i \in Z_n$ and sets $\sigma_L^i = z_L^i$.
  - with the remaining probability $1 - 1/(2p)$, the dealer chooses σ_L^i as in Case I described in Section 4.1.

Similar changes are made in the protocol without the dealer.

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 larger, 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.

5 The 3-party Protocol Tolerating Two Corrupt Parties

In this section we describe an r-round 3-party protocol tolerating two corrupt parties. Unlike Section 4, we directly describe our protocol without any dealer. The formal description of the 3-party protocol, Protocol MPCFor3Protocol$_r$, appears in Figure 1 and Figure 2.

We next sketch the ideas of the protocol. As in all our protocols, we construct a protocol with two phases. The first phase is a preliminary phase in which the parties compute a given functionality (securely-with-abort with cheat detection). The output of this functionality for party p_j includes the messages that p_j broadcasts throughout the second phase – called the interaction phase. For simplicity of presentation, in the rest of the paper, we assume that in the interaction phase of the protocol the adversary is a *fail-stop* adversary. That is, all parties follow the protocol with one exception: the corrupt parties may abort the computation at any time. For our protocols, this assumption is without loss of generality, since in each round there is a small number of messages that each party can send. We have already demonstrated how to limit the adversary to aborts in this case by signing (in the preprocessing phase) any such possible message. Using this assumption, we can omit the discussion regarding signing of the messages.

In the preliminary phase of the protocol, for each round i and for each subset $L \subset \{1,2,3\}$ of size one or two a value σ_L^i is chosen similarly to the way the values in the protocols in Section 4 are chosen; this value is used by the parties $\{p_j : j \in L\}$ if the other party/parties abort in round $i+1$. Specifically, there is a special round, called $i^\star$, chosen with uniform distribution from $\{1,\ldots,r\}$. Prior to round $i^\star$, the values chosen for each subset depends only on the inputs of the subset: random inputs are chosen for the parties not in the subset and the function f_n is computed with the inputs of the subset and the random inputs for other party/parties. Starting in round $i^\star$, the value of each subset is the output w of f_n on the inputs of all parties.

If no party aborts during the protocol, then each party p_j outputs the value $\sigma_{\{j\}}^r$. If two corrupt parties abort in some round i, then the third party p_j outputs the value $\sigma_{\{j\}}^{i-1}$. The difficult case is when one party, say p_3, aborts in some round i. In this case one of the active parties p_1, p_2 might be corrupt. Thus, the parties execute a variant of the two-party r-round $O(1/r)$-secure protocol of [18] to compute f_n. Specifically, if $i \geq i^\star$, then in each round of the two-party protocol the parties get the value w (thus, an abort of p_3 after round $i^\star$ does not affect the output). If $i < i^\star$, then we would like to execute the following protocol: (1) a new special round $i^\star_{\{1,2\},i}$ is selected with uniform distribution from $\{1,\ldots,r\}$, and (2) an r-round protocol is executed, where prior to round $i^\star_{\{1,2\},i}$ each party gets a value that depends only on its input and starting from round $i^\star_{\{1,2\},i}$, each party gets $\sigma_{\{1,2\}}^{i-1}$.

The protocol sketched above is flawed: suppose now that p_1, p_2 are corrupt. In each round i they can simulate the execution of the two-party protocol that they would have executed if p_3 has aborted. The first round i in which *all* the values

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

Computing default values:

1. Set $w \leftarrow f_n(x_1, x_2, x_3)$ and select $i^\star \in \{1, \ldots, r\}$ with uniform distribution.
2. For each $1 \leq i < i^\star$ and for each $j \in \{1, 2, 3\}$,
   (a) Set $L = \{1, 2, 3\} \setminus \{j\}$,
   (b) With probability $1/\sqrt{r}$, set $i^\star_{L,i} = 1$. With the remaining probability, select $i^\star_{L,i} \in \{1, \ldots, r\}$ with uniform distribution.
   (c) Select uniformly at random $\widehat{z}_j \in X_n$, for each $\ell \in L$ set $\widehat{z}_\ell = x_\ell$, and set $\sigma^i_L \leftarrow f_n(\widehat{z}_1, \widehat{z}_2, \widehat{z}_3)$.
   (d) For each $1 \leq i_2 < i^\star_{L,i}$ and for each $\ell \in L$,
      i. Set $\widehat{z}_\ell = x_\ell$ and for each $j \in \{1, 2, 3\} \setminus \{j\}$, select $\widehat{z}_j$ with uniform distribution from X_n.
      ii. Set $\sigma^i_{L,\ell,i_2} \leftarrow f_n(\widehat{z}_1, \widehat{z}_2, \widehat{z}_3)$.
   (e) For each $i^\star_{L,i} \leq i_2 \leq r$ and for each $\ell \in \{1, 2, 3\} \setminus \{j\}$, set $\sigma^i_{L,\ell,i_2} = \sigma^i_L$.
3. For each $i^\star \leq i < r$, for each $j \in \{1, 2, 3\}$, for each $1 \leq i_2 \leq r$, set $L = \{1, 2, 3\} \setminus \{j\}$ and for each $\ell \in L$ set $\sigma^i_{L,\ell,i_2} = w$.
4. For each $1 \leq i < i^\star$ and for each $j \in \{1, 2, 3\}$,
   (a) Set $\widehat{x}_j = x_j$ and for each $\ell \in \{1, 2, 3\} \setminus \{j\}$, select uniformly at random $\widehat{x}_\ell \in X_n$.
   (b) Set $\sigma^i_j \leftarrow f_n(\widehat{x}_1, \widehat{x}_2, \widehat{x}_3)$.
5. For each $i^\star \leq i \leq r$ and for each $j \in \{1, 2, 3\}$, set $\sigma^i_j = w$.

Computing messages:

1. For each $1 \leq i \leq r$, each $1 \leq i_2 \leq r$, each $L \subset \{1, 2, 3\}$ s.t. $|L| = 2$, and each $\ell \in L$, share σ^i_{L,ℓ,i_2} in a $|2|$-out-of-$|2|$ secret-sharing scheme for the parties $\{p_j : j \in L\}$.
   For each $j \in L$, let $S^i_{L,\ell,i_2,j}$ be the share of p_j of the secret σ^i_{L,ℓ,i_2}.
2. For each $1 \leq i \leq r$ and for each $j \in \{1, 2, 3\}$,
   (a) Set $m_{i,j} = \sigma^i_j \circ \left(S^i_{\{j,\ell\},\ell,i_2,j}\right)_{1 \leq i_2 \leq r; \ell \in \{1,2,3\} \setminus \{j\}}$.
   (b) Share $m_{i,j}$ in a $|3|$-out-of-$|3|$ secret-scheme. For each $\ell \in \{1, 2, 3\}$, let $S_{i,j,\ell}$ be the share of p_ℓ of the secret $m_{i,j}$.
   (c) For each $j \in \{1, 2, 3\}$ compute $M_{i,j} \leftarrow (S_{i,j,\ell})_{\ell \in \{1,2,3\} \setminus \{j\}}$.

Outputs: Each party p_j receives

- The messages $M_{1,j}, \ldots, M_{r,j}$ that p_j broadcasts during the protocol.
- p_j's shares $S_{1,j,j}, \ldots, S_{r,j,j}$ for reconstructing the values for the three party protocol.
- p_j's shares $S^i_{L,j,i_2,j}$ for each $1 \leq i \leq r$, each $1 \leq i_2 \leq r$, each $L \subset \{1, 2, 3\}$ s.t. $|L| = 2$, and $j \in L$ for reconstructing the values in the two-party protocol for $\{p_\ell : \ell \in L\}$.

Fig. 1. Functionality ShareGenFor3$_r$.

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

Preliminary phase:

1. The parties execute a secure-with-abort and cheat-detection protocol computing Functionality ShareGenFor3$_r$. Each honest party p_j inputs y_j as its input for the functionality.
2. If an abort has occurred, that is, the output of the honest parties is "`abort`$_j$" for at least one index j, then, for each such index j, the remaining active parties mark p_j as inactive, i.e., set $D = D \cup \{j\}$ and execute the instructions for one or two active parties with $i = 1$.
3. Else (no party has aborted), denote $D = \emptyset$ and proceed to the first round.

Instructions for three active parties:

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

(a) Each party p_j broadcasts $M_{i,j}$.

(b) For every p_j that aborts, all parties mark p_j as inactive and the active parties execute the instructions for one or two active parties.

`At the end of round` r`:` Each active party p_j reconstructs the value σ_j^r, outputs it, and halts.

Instructions for two active parties indexed by L:

(a) If $i = 1$, then execute the two party protocol of [18] for the functionality $f_n(\cdot,\cdot,\cdot)$ in which, the input for the aborted party is selected uniformly at random from X_n and halt.

(b) Else, each active party p_j reconstructs $m_{i-1,j}$ and for each round $i_2 = 1, \ldots, r$ in the two-party protocol:

i. Each active party p_j where $j \in L$ broadcasts the share $S^{i-1}_{L,\ell,i_2,j}$ where p_ℓ is the other active party.

ii. If p_j aborts, then the remaining active party p_ℓ marks p_j as inactive and

- If $i_2 > 1$, i.e., at least one round of the two-party protocol was competed, then, p_ℓ reconstructs $\sigma^{i-1}_{L,\ell,i_2-1}$, outputs it, and halts.
- Else, if no round of the two-party protocol was competed, then, the active party p_ℓ reconstructs the value σ_ℓ^{i-1} from the three party protocol, outputs it, and halts.

(c) At the end of round r: Each active party p_j reconstructs the value $\sigma^{i-1}_{L,j,r}$, outputs it, and halts.

Instructions for one active party p_ℓ:

(a) Set $\widehat{x}_\ell = y_\ell$ and for every $j \in D$, select $\widehat{x}_j$ with uniform distribution from X_n.

(b) Set $w \leftarrow f_n(\widehat{x}_1, \widehat{x}_2, \widehat{x}_3)$, output w, and halt.

Fig. 2. The 3-party protocol MPCFor3$_r$ for computing $\mathcal{F}$.

they get in the simulated protocol are equal is $i^\star$. Thus, they can determine $i^\star$ and bias the output of the protocol with a high probability. To overcome this problem, we modify the way that $i^\star_{L,i}$ is chosen prior to round i: with probability $O(1/\sqrt{r})$ set $i^\star_{L,i} = 1$, and with the remaining probability choose it at random from $\{1, \ldots, r\}$. Notice that the simulated protocol in the case $i^\star_{L,i} = 1$ looks like the simulated protocols in rounds starting from $i^\star$, thus, the probability that the corrupted parties guess $i^\star$ is $O(\sqrt{r}/r) = O(1/\sqrt{r})$. However, a corrupt p_2 can bias the protocol by guessing $i^\star_{L,i} = 1$ and aborting in round 1 of the two-party protocol. This can cause an additional bias of at most $O(1/\sqrt{r})$. All together, the resulting protocol is $O(1/\sqrt{r})$-secure.

We next explain how the two-party protocol is executed. The two-party protocol of [18] has, again, two stages: a preliminary stage and an interaction phase. In our protocol, we have only one preliminary stage, in which all preliminary phases of the two-party protocols are executed simultaneously. That is, in the preliminary phase, for every round $1 \leq i \leq r$ and for every $L \subset \{1, 2, 3\}$ of size two, the preliminary phase of the two-party protocol of [18] is executed for L (using σ^i_L and $i^\star_{L,i}$). Let $(S^i_{L,j})_{j \in L}$ be the two outputs of the preliminary phase that should be given to the parties indexed by L. Each $S^i_{L,j}$ for $j \in L$ is shared using a 3-out-of-3 secret sharing scheme. The output of the preliminary phase of each party includes exactly one of these shares.

Later, in each interaction round i, for each $L \subset \{1, 2, 3\}$ of size two and for each $j \in L$, the parties p_k, where $k \neq j$, broadcast their shares of $S^i_{L,j}$. Thus, p_j obtains $S^i_{L,j}$ while the other two parties learn nothing on it. Now, if a party, say p_3, aborts in round i, parties p_1 and p_2 can execute the two party protocol of round $i-1$ using $S^{i-1}_{\{1,2\},1}$ and $S^{i-1}_{\{1,2\},2}$ respectively.

In the above, we only sketched the protocols. The formal description of the functionality computed by the preliminary phase appears in Figure 1 and the protocol appears in Figure 2. The proof that the protocol is 1/p-secure will appear in the full version of the paper. To construct a 3-party protocol for functionalities where the size of range is small we use the same trick used in Section 4.3: With some small probability a value given to a set is chosen from the range prior to $i^\star$ in the 3-party interaction and prior to $i^\star_{L,i}$ in the two parties' protocols. The m-party protocols tolerating up to $m-1$ corrupt parties, uses the same ideas as our 3-party protocols. In a preliminary phase, $i^\star$ and values σ^i_L are chosen as above. If one party aborts in some round i, then the remaining $m-1$ parties execute our $(m-1)$-party protocol, where if $i \geq i^\star$ then it uses $i^\star_{L,i} = 1$, and if $i < i^\star$ then $i^\star_{L,i} = 1$ with some probability and $i^\star_{L,i}$ is random otherwise. In this $(m-1)$-party protocol, if a party aborts the remaining $m-2$ parties execute our $(m-2)$-party protocol (again with its special round being set to 1 with some probability), and so on.

6 Best of Both Worlds – The 1/p Way

We study the question of whether or not it is possible to construct "best of both worlds" protocols, when the fall-back security guarantee is 1/p-security

or $1/p$-security-with-abort. We investigate whether protocols with these weaker notions of security are possible when full privacy cannot be guaranteed. In the full version of the paper, we construct protocols that guarantee full-security whenever less than half of the parties are corrupt, and $1/p$-security-with-abort otherwise. These protocols are simpler and more efficient than the protocols that guarantee fall-back $1/p$-(full)-security, which we describe below.

To construct the protocols that have fall-back $1/p$-(full)-security, we show in Section 6.1 how to transform $1/p$-secure protocols of a certain type into protocols that retain the same security for the case of no honest majority, while guaranteeing full-security whenever less than half of the parties executing the protocol are corrupt. Specifically, we will prove the following theorem.

Theorem 3. *Let $\mathcal{F}$ be an m-party (possibly randomized) functionality. If enhanced trap-door permutations exist, and if m is constant and the size of the domain $g(n)$ and the size of the range $g(n)$ are bounded by a polynomial in the security parameter n, then for any polynomial $p(n)$ there is an $r(n)$-round $1/p(n)$-secure protocol computing $\mathcal{F}$ tolerating up to $m-1$ corrupt parties and, in addition, guarantees full-security in the presence of an honest majority, where* $r(n) = 2 \cdot p(n)^{2^m} \cdot \Big(d(n) \cdot g(n)\Big)^{2^{O(m)}}$.

A similar theorem will appear in the full version of the paper for the result of applying the above transformation to the protocol implying the second item of Theorem 2. The resulting protocol has polynomially many rounds even when the number of parties is $\frac{1}{2}\log\log n$ provided that the functionality is deterministic and the size of the domain of inputs is constant.

6.1 Best of Both Worlds – The $1/p$-(full)-Security Variant

In this section we show how to transform $1/p$-secure protocols of a certain type into protocols that retain the security of the original protocol for the case of no honest majority, while guaranteeing full-security whenever less than half of the parties executing the protocol are corrupted. Intuitively, the transformation works if the original protocol has full security against a weaker adversary that can only abort at the beginning of each round (i.e., before seeing the messages of the honest parties for this round). Specifically, this transformation can be applied to all protocols in this paper that have full correctness (namely, the protocols that assume that the sizes of the domain and the range are polynomial). Note that protocols that do not have full correctness (at least for the case of honest majority) do not guarantee full-security for the case of honest majority. At the end of this section, we will hint why the resulting protocols guarantee the desired security notion. The full argument will appear in the full version of the paper.

The basic structure of protocols that can be transformed. For simplicity of presentation we first present our transformation for an (original) protocol with a certain structure. Consider an m-party protocol for computing a functionality $\mathcal{F}$

that has the following structure: The interaction starts with a preliminary phase in which the parties execute a secure-with-abort with cheat-detection protocol for computing the messages that the parties are to send in the next r interaction rounds; after this phase, each party p_j holds a (signed) message M_j^i for each round $1 \leq i \leq r$. In each interaction round i, each party p_j broadcasts the message M_j^i. Any failure of party p_j to broadcast the signed message as prescribed by the protocol is considered as an abort of p_j. The adversary can cause the protocol to prematurely terminate by instructing some $t_A < \lceil \frac{m}{2} \rceil$ corrupted parties to abort. Unless premature termination takes place, the protocol proceeds normally (that is, as long as less than t_A of parties have aborted). In the case of premature termination, the remaining parties engage in a protocol Π_{TERM} for agreeing on the output of the protocol, based on the view of the parties in the protocol so far. More specifically, the decision upon the output is based on the outputs of the (remaining) parties from the preliminary phase, on the messages broadcast until round $i-1$, and on the set of parties that have aborted D.

Indeed, all our protocols that were described in previous sections have the above structure. For the sake of being concrete, however, in the following we will describe the transformation as applied to the protocol of Section 4.2. In this protocol Π_{TERM} is a protocol for reconstructing the output that is always executed with a guaranteed honest majority.

The transformation. The core of the change is a mask we add to the messages of the parties in each round. This mask is shared in a $(\lfloor \frac{m}{2} \rfloor + 1)$-out-of-$m$ secret-sharing scheme. Hence, the messages of the parties disclose the original messages if and only if a majority of the parties work together to reconstruct the appropriate masks. Below we explain this change in more detail.

Denote by M_j^i the message that party p_j is instructed to broadcast in round i of the original protocol. That is, the output of party p_j from the preliminary phase of the original protocol includes the messages $M_j^1, \ldots, M_j^r$. In the preliminary phase of the new protocol a random string r_j^i will be selected for each party p_j and each round i, and the sequence $\hat{M}_j^1, \ldots, \hat{M}_j^r$ will be given to party p_j, where $\hat{M}_j^i = M_j^i \oplus r_j^i$. In addition, each party will also receive a share of r_j^i in a $(\lfloor \frac{m}{2} \rfloor + 1)$-out-of-$m$ Shamir secret-sharing scheme. The message $\hat{M}_j^i$ and the shares of its mask r_j^i are all signed.

Each interaction round of the original protocol is turned into a two-phased round in the new protocol. In the first phase, each party p_j broadcasts the message $\hat{M}_j^i$. In the second phase, the parties reconstruct all masks of round i by broadcasting all shares of masks r_j^i, for $1 \leq j \leq m$. If both phases are completed, then the parties have the same information as in the original protocol. Any failure of party p_j to broadcast the signed message as prescribed by the protocol is considered as an abort of p_j (including messages added by the transformation).

The adversary can cause the protocol to prematurely terminate only by instructing some $t_A < \lceil \frac{m}{2} \rceil$ corrupted parties to abort. We handle such premature termination in round i by instructing the parties to behave as if premature termination has occurred at the beginning or at the end of round i (i.e., at the

beginning of round i+1). Specifically, if premature termination takes place before the reconstruction of the masks (in the second phase of round i) is completed, then the remaining parties will behave as if the original protocol was terminated at the beginning of round i. That is, they will engage in a protocol Π_{TERM} for agreeing on the output of the protocol, based on the messages broadcast until round $i-1$ and on the set of parties that have aborted D. Otherwise, if the reconstruction of the masks was completed before the abort, then the remaining parties will behave as if the original protocol was terminated at the beginning of round $i+1$.

The security of the new protocol. In the full version of this paper, we argue that applying the above transformation to any of the our protocols that assume that the domain and the range are polynomial, results in a protocol that is (i) fully secure against a malicious adversary that can corrupt any strict minority of the parties, and (ii) $1/p$-secure against a malicious adversary that can corrupt up to t parties. Furthermore, we will show that this is true for any protocol that has the structure defined above and, in addition, satisfies a few simple requirements.

We now give some intuition for why this is true if the transformation is applied to the protocol of Section 4.2. We need to consider two cases. In the case that at list half of the parties are malicious, it is quite straightforward to see that the adversary attacking the transformed protocol is not any more powerful than an adversary for the original protocol, since once the adversary sees the messages of the corrupted parties, the masks add no new information.

In the case of an honest majority, the shares of r_j^i that the corrupted parties see, do not reveal anything to the adversary as long as the shares of honest parties are not revealed (these shares are only revealed in the second phase of round i). Thus, if the adversary causes a premature termination during the first phase of round i, then it has no more information than is obtained in the original protocol (by an adversary corrupting the same subset of parties) until the beginning of round i. If it aborts after the first phase, then the honest parties will succeed in reconstructing the masks. Thus, the adversary is no more powerful then an adversary for the original protocol that can only abort at the beginning of each round. However, the security of the original protocol can only be violated if the adversary causes premature termination during round $i^\star$. Finally, the reconstruction is fully secure in the presence of an honest majority.

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

For deterministic functions, our protocols are efficient when the number of parties m is constant and the size of the domain or range is at most polynomial (in the security parameter n) or when the number of parties is $\log\log n$ and the size of the domain is constant. We 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. That is, we prove the following two theorems.

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.*

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.*

7.1 Impossibility of Achieving "The Best of Both Worlds" for General Functionalities

Above we showed that $1/p$-secure computation is impossible in general 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. Since a "Best of Both Worlds" type protocol with fall-back $1/p$-security is in particular $1/p$-secure, the same impossibility results are implied for protocols of this type (i.e., guaranteeing full-security with an honest majority and $1/p$-security otherwise). We show that such protocols are impossible in general, even when allowing the fall-back security to be the weaker notion of $1/p$-security-with-abort. Hence, we show that the results discussed in Section 6 are somewhat optimal.

We start by showing in that for general functionalities (i.e., where both domains and both ranges may be super-polynomial), it is impossible to construct even 3-party protocols that simultaneously achieve full-security for the case of honest majority (i.e., at most one corrupted party) and $1/p$-security-with-abort with no honest majority. We then use this result to prove general impossibility results, that is, to prove the two following theorems:

Theorem 6. *For every $m(n) = \omega(\log n)$, there exists a deterministic $m(n)$-party functionality $\mathcal{F}'$ with domain $\{0,1\}$ that cannot be computed simultaneously guaranteeing full-security with an honest majority and $1/p$-security-with-abort for $p \geq 2 + 1/\operatorname{poly}(n)$ against an adversary controlling $\lfloor m(n)/2 \rfloor + 1$ parties.*

Theorem 7. *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 computed simultaneously guaranteeing full-security with an honest majority and $1/p$-security-with-abort for $p \geq 2 + 1/\operatorname{poly}(n)$ against an adversary controlling $\lfloor m(n)/2 \rfloor + 1$ parties.*

References

[1] D. Beaver and S. Goldwasser. Multiparty computation with faulty majority. In *30th FOCS*, pages 468–473, 1989.
[2] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *22nd STOC*, pages 503–513, 1990.
[3] A. Beimel, E. Omri, and I. Orlov. Protocols for multiparty coin toss with dishonest majority. In *CRYPTO 2010*, volume 6223 of *LNCS*, pages 538–557, 2010.
[4] M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest. A fair protocol for signing contracts. In *12th ICALP*, pages 43–52, 1985.

[5] M. Blum. How to exchange (secret) keys. *ACM Trans. Comput. Syst.*, 1(2):175–193, 1983.
[6] D. Boneh and M. Naor. Timed commitments. In *CRYPTO 2000*, volume 1880 of *LNCS*, pages 236–254, 2000.
[7] R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *18th STOC*, pages 364–369, 1986.
[8] R. Cleve. Controlled gradual disclosure schemes for random bits and their applications. In *CRYPTO '89*, volume 435 of *LNCS*, pages 573–588, 1990.
[9] I. Damgård. Practical and provably secure release of a secret and exchange of signatures. *J. of Cryptology*, 8(4):201–222, 1995.
[10] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *CACM*, 28(6):637–647, 1985.
[11] Z. Galil, S. Haber, and M. Yung. Cryptographic computation: Secure fault-tolerant protocols and the public-key model. In *CRYPTO '87*, volume 293 of *LNCS*, pages 135–155, 1988.
[12] J. A. Garay, P. D. MacKenzie, M. Prabhakaran, and K. Yang. Resource fairness and composability of cryptographic protocols. In *TCC 2006*, volume 3876 of *LNCS*, pages 404–428, 2006.
[13] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *19th STOC*, pages 218–229, 1987.
[14] S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In *CRYPTO '90*, volume 537 of *LNCS*, pages 77–93, 1991.
[15] S. D. Gordon, C. Hazay, J. Katz, and Y. Lindell. Complete fairness in secure two-party computation. In *40th STOC*, pages 413–422, 2008.
[16] S. D. Gordon, Y. Ishai, T. Moran, R. Ostrovsky, and A. Sahai. On complete primitives for fairness. In *TCC 2010*, volume 5978 of *LNCS*, pages 91–108, 2010.
[17] S. D. Gordon and J. Katz. Complete fairness in multi-party computation without an honest majority. In *TCC 2009*, pages 19–35, Berlin, Heidelberg, 2009.
[18] S. D. Gordon and J. Katz. Partial fairness in secure two-party computation. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 157–176, 2010.
[19] Y. Ishai, J. Katz, E. Kushilevitz, Y. Lindell, and E. Petrank. On achieving the "best of both world" in secure multiparty computation. *SIAM J. on Computing*, 40(1), 2011. Journal version of [20, 21].
[20] Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In *CRYPTO 2006*, number 4117 in LNCS, pages 483–500, 2006.
[21] J. Katz. On achieving the "best of both worlds" in secure multiparty computation. In *39th STOC*, pages 11–20, 2007.
[22] M. Luby, S. Micali, and C. Rackoff. How to simultaneously exchange a secret bit by flipping a symmetrically-biased coin. In *24th FOCS*, pages 11–21, 1983.
[23] T. Moran, M. Naor, and G. Segev. An optimally fair coin toss. In *TCC 2009*, pages 1–18, 2009.
[24] R. Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *36th STOC*, pages 232–241, 2004.
[25] B. Pinkas. Fair secure two-party computation. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 87–105, 2003.
[26] A. C. Yao. How to generate and exchange secrets. In *27th FOCS*, pages 162–167, 1986.