Almost-Optimally Fair Multiparty Coin-Tossing with Nearly Three-Quarters Malicious*

Bar Alon and Eran Omri

Department of Computer Science, Ariel University
alonbar08@gmail.com, omrier@ariel.ac.il

Abstract. An α-fair coin-tossing protocol allows a set of mutually distrustful parties to generate a uniform bit, such that no efficient adversary can bias the output bit by more than α. Cleve [STOC 1986] has shown that if half of the parties can be corrupted, then, no r-round coin-tossing protocol is $o(1/r)$-fair. For over two decades the best known m-party protocols, tolerating up to $t \geq m/2$ corrupted parties, were only $O\left(t/\sqrt{r}\right)$-fair. In a surprising result, Moran, Naor, and Segev [TCC 2009] constructed an r-round two-party $O(1/r)$-fair coin-tossing protocol, i.e., an optimally fair protocol. Beimel, Omri, and Orlov [Crypto 2010] extended the result of Moran et al. to the *multiparty setting* where strictly fewer than 2/3 of the parties are corrupted. They constructed a $2^{2^k}/r$-fair r-round m-party protocol, tolerating up to $t = \frac{m+k}{2}$ corrupted parties.

Recently, in a breakthrough result, Haitner and Tsfadia [STOC 2014] constructed an $O\left(\log^3(r)/r\right)$-fair (almost optimal) three-party coin-tossing protocol. Their work brought forth a combination of novel techniques for coping with the difficulties of constructing fair coin-tossing protocols. Still, the best coin-tossing protocols for the case where more than 2/3 of the parties may be corrupted (and even when $t = 2m/3$, where $m > 3$) were $\theta\left(1/\sqrt{r}\right)$-fair. We construct an $O\left(\log^3(r)/r\right)$-fair m-party coin-tossing protocol, tolerating up to t corrupted parties, whenever m is constant and $t < 3m/4$.

1 Introduction

Secure multiparty computation allows a set of mutually distrustful parties to perform a computational task, while guaranteeing some security properties to hold. Examples of desirable security properties of a secure protocol are correctness, privacy, and *fairness* (roughly, the requirement that either all parties receive their respective outputs, or none do). When a strict majority of honest parties can be guaranteed, protocols for secure computation (see, e.g., [19, 9]) provide full security, i.e., they provide all the security properties mentioned above (and others), including fairness. When there is no honest majority, however, this is no longer the case, and full security (specifically, full fairness) is not achievable in

*Research supported by ISF grant 544/13.

general. As was shown by Cleve [14], this is already evident for the elementary (no input) task of coin-tossing.

The coin-tossing functionality, introduced by Blum [12], allows a set of parties to agree on a uniformly chosen bit. Cleve [14] showed that this functionality cannot be computed with complete fairness without a strict honest majority. He proved that for any r-round two-party coin-tossing protocol, there exists an (efficient) adversary that can bias the output of the honest party by $\Omega(1/r)$. Cleve's impossibility naturally generalizes to the multiparty setting with no honest majority and has ramifications to general secure computation, implying that any function that implies coin-tossing (e.g., the XOR function) cannot be computed with full fairness without an honest majority. The question of optimal fairness for the coin-tossing functionality seems to be crucial towards understanding general secure and fair multiparty computation.

On the positive end, Averbuch et al. [6], Cleve [14] showed how to compute the coin-tossing functionality with partial fairness, limiting the bias of any adversary to $O\left(1/\sqrt{r}\right)$. For over two decades, these constructions were believed to be optimal. This belief was supported by the work of Cleve and Impagliazzo [15], showing that in a model, where commitments are available only as black-box (and no other assumptions are made), the bias of any coin-tossing protocol is $\Omega(1/\sqrt{r})$. In a breakthrough result, Moran, Naor, and Segev [30] showed that the $\Omega(1/r)$-bias lowerbound of Cleve is tight for the case of two-party coin-tossing. They constructed an r-round two-party coin-tossing protocol with bias $O(1/r)$. The protocol of Moran et al. follows the special-round paradigm[1], previously appearing in [27, 22].

Beimel, Omri, and Orlov [8] constructed (via the special-round paradigm) an optimal $O(1/r)$-bias protocol for any constant number of parties, whenever strictly less than a 2/3-fraction of the parties are malicious. More accurately, for their construction to yield an $O(1/r)$ bound on the bias of their protocol, it suffices that the gap between the number of corrupted parties and the number of honest parties is constant (rather than the total number of parties).

Still, the question whether optimal $O(1/r)$-coin-tossing was possible when the set of malicious parties may consist of two-thirds or more of the parties remained open. Specifically, even the case of three-party optimally-fair coin-tossing, where two of the parties may be corrupted remained unsettled. Answering the question regarding the three party case seemed to require new techniques and a novel understanding of coin-tossing protocols. In another breakthrough result, Haitner and Tsfadia [24] constructed an $O\left(\log^3(r)/r\right)$-fair (almost optimal) three-party coin-tossing protocol. Their work, indeed, offers some profound insight into the difficulties of constructing coin-tossing protocols, and brings forth a combination of novel techniques for coping with these difficulties. However, while it may be tempting to expect that the solution for the three-party case (and, specifically, that of [24]) will soon lead to a solution for fair coin-tossing for any (constant) number of parties, this has not been the case so far.

[1] The idea is to randomly and secretly choose a special round in which the parties unknowingly get the output of the computation.

1.1 Our results

Our main contribution is a multiparty coin-tossing protocol that has small bias whenever the number of parties is constant fewer than 3/4 of them are corrupted.

Theorem 1 (informal). *Assume that oblivious transfer protocols exist. Let m and t be* constants *(in the security parameter n) such that $m/2 \leq t < 3m/4$, and let $r = r(n)$ be an integer. There exists an r-round m-party coin-tossing protocol tolerating up to t corrupted parties that has bias $O(2^{2^m} \log^3(r)/r)$.*

The formal statements and proofs implying Theorem 1 are given in Section 3, a warmup construction illustrating the ideas behind the general construction is given in Section 1.4. The 2^{2^m} factor in the upperbound on the bias of our construction is due the fact that in each round, the adversary sees defense values for many corrupted subsets. For this reason, we require m to be constant.

1.2 Additional Related Work

Partially fair coin-tossing is an example of $1/p$-secure computation. Informally, a protocol is $1/p$-secure if it emulates the ideal functionality within $1/p$ distance. The formal definition of $1/p$-secure computation appears in Section 2.3.1. $1/p$-security *with abort* was suggested by Katz [27]. Gordon and Katz [21] defined $1/p$-security and constructed 2-party $1/p$-secure protocols for every functionality whose size of either the domain or the range of the functionality is polynomial (in the security parameter). Beimel et al. [7] studied multiparty $1/p$-secure protocols for general functionalities. The main result in [7] is constructions of $1/p$-secure protocols that are resilient against *any* number of corrupted parties, provided that the number of parties is constant and that the size of the range of the functionality is at most polynomial in the security parameter n. The bias of the coin-tossing protocol resulting from [7] is $O(1/\sqrt{r})$.

The impossibility result of Cleve [14] made many researchers believe that no interesting functions can be computed with full fairness without an honest majority. A surprising result by Gordon et al. [22] showed that there are even functions containing embedded XOR that can be computed with fairness. This led to a line of works, investigating complete fairness in secure multiparty computation without an honest majority [3, 2, 29]. Recently, Asharov et al. [4] gave a full characterization of fairness secure two-party computation of Boolean functions.

Coin-tossing is an interesting and useful task even in weaker models, e.g., secure-with-abort coin-tossing – where honest parties are not requested to output a bit upon a premature abort by the adversary, and weak coin-tossing – where each party has an a priori desire for the output bit. Indeed, the latter type of coin-tossing was the one formulated by Blum [11], who suggested a fully secure weak (and actually, secure with abort) coin-tossing protocol based on the existence of one-way functions ([25, 31]). His protocol is also a 1/4-secure implementation of the fair coin-tossing functionality. Conversely, the existence of

secure-with-abort protocols imply the existence of one-way functions [28, 23, 10]. For the cryptographic complexity of optimally-fair coin-tossing, [16, 17] gave some evidence that one-way functions may not suffice.

1.3 Our Techniques

Towards explaining the ideas behind our protocol, we give a brief overview of the constructions of [30, 8, 24]. We restrict our discussion to the fail-stop model, where corrupted parties follow the prescribed protocol, unless choosing to prematurely abort at some point in the execution. Indeed, the core difficulties in constructing fair coin-tossing protocols stand in this model as well. Specifically, an r-round multiparty coin-tossing protocol in the fail-stop model can be adapted to the malicious setting by adding signatures to each message (or by applying the GMW compiler [19]).

1.3.1 The Protocol of Moran et al. [30]. The protocol of Moran, Naor, and Segev [30] is a two-party r-round coin-tossing protocol with optimal bias $1/4r$. That is, their protocol matches the lowerbound of Cleve [14] (up to a factor 2). The basic idea of the protocol is that in each round i, each of the parties is given an independently chosen uniform bit, which will be its output, in case the other party aborts. This is done until some special round i^*. From round i^* and on, both parties get the same bit c. Finally, i^* is chosen uniformly from $[r]$ and is kept secret from the parties. The security of the protocol relies on the inability of the adversary to guess the value of i^* with probability higher than $1/r$. We next give a slightly more detailed overview of the MNS protocol restricted to fail-stop adversaries.

A skeleton for two-party coin-tossing protocols. We start by describing the skeleton for the two-party protocol of [30]. Indeed, this is a more generic skeleton and can be used to describe any two-party coin-tossing protocol (A, B).

The preliminary phase of the protocol. In this phase, the parties jointly compute defense values for each of the r rounds of interaction. Denote the defense value assigned to A for round $i \in [r]$ by a_i and the value assigned to B for round i by b_i (in the MNS protocol, these defense values are actually bits). At the end of this preliminary phase, the parties do not learn these defense values, but rather hold a share in a 2-out-of-2 secret sharing scheme (separately, for each defense value). Denote by $a_i[\mathrm{P}]$ and $b_i[\mathrm{P}]$ the shares of a_i and b_i (respectively) held by party P.

Interaction rounds. In round i, party A reveals $b_i[\mathsf{A}]$ and party B reveals $a_i[\mathsf{B}]$. Specifically, in round i, party A learns a_i and party B learns b_i. The role of these defense values is to define the output of an honest party, upon a premature abort of the other party. For example, if party A aborts in round i (not allowing B to learn b_i), then B halts and outputs b_{i-1}. If an abort never occurs, then parties output $a_r = b_r$.

The MNS instantiation of the two-party skeleton. We now specify how the defense values are selected in the protocol of [30]. The parties jointly select a special round number $i^* \in \{1, \ldots, r\}$, uniformly at random, and select bits $a_1, \ldots, a_{i^*-1}, b_1, \ldots, b_{i^*-1}$, independently, uniformly at random. Then, they uniformly select a bit $w \in \{0, 1\}$ and set $a_i = b_i = w$ for all $i^* \leq i \leq r$.

The security of the protocol follows from the fact that, unless the adversary aborts in round i^*, it cannot bias the output of the protocol. This is true, since before round i^* the view of the adversary is independent of the prescribed output bit w, and hence, given that the adversary aborts *before* round i^*, the output of the honest party is a uniform bit. On the other hand, after round i^* is completed, the output of the honest party is fixed. Hence, aborting in any round after i^* is equivalent to never aborting at all, therefore, given that the adversary aborts *after* round i^*, the output of the honest party is also a uniform bit. Finally, the view of any of the parties up to round $i \leq i^*$ is independent of the value of i^*, hence, any adversary corrupting a single party can guess i^* with probability at most $1/r$.

1.3.2 The Protocols of Haitner and Tsfadia [24]. Haitner and Tsfadia [24] constructed a three-party r-round coin-tossing protocol with close to optimal bias $O\left(\log^3 r/r\right)$. Towards achieving this goal, Haitner and Tsfadia [24] first constructed several new two-party fair coin-tossing protocols with bias $O\left(\log^3 r/r\right)$. Evidently, the bias of these protocols does not match the Cleve [14] lowerbound (as does the MNS protocol), however, the techniques and insight introduced in these constructions make them interesting even before considering the final three-party construction, for which they serve as a building block. In fact, most of the techniques that enable the three-party construction of [24] come up already in their two-party protocols.

Before describing the protocols of [24], let us first highlight some of the ideas underlying them. We stress that none of their protocols follows the special round paradigm. Alternatively, their protocols have the value of the game (i.e., the expected outcome in an honest continuation of the current state) gradually shift from being $1/2$ (or some other $\alpha \in [0, 1]$, for that matter) to being either 0 or 1. This is done by having the parties run in the background – jointly and hidden from each of them – a protocol with a gradually shifting and publicly known game value (in this case, a weighted variant of the majority protocol of [6, 14]). Let O_i be the game value in round i.

One of the core observations underlying all the constructions of Haitner and Tsfadia [24] is that letting the defense value a_i be a bit sampled according to O_i, fully protects A in case of an abort by B in round i. More importantly, if the gap between O_i and O_{i-1} is typically $O\left(1/\sqrt{r}\right)$, then a_i does not reveal too much information about the current value of O_i to A. Finally, Haitner and Tsfadia [24] show that a_i can be instantiated, not only as a bit, but also as a description of a full execution of a two-party protocol with output and (defense values) sampled according to O_i (where, this form of a_i still does not reveal too much information

about the current value of O_i to A). Going from here to their construction of a three-party coin-tossing protocol is fairly natural.

We next describe the two-party protocols of Haitner and Tsfadia [24]. We do so using the skeleton for two-party protocols described in Section 1.3.1. That is, we explain how the defense values a_i, b_i for each round i are selected. We note that Haitner and Tsfadia [24] did not present their protocols in this exact manner, but rather divided each interaction round i into two steps. The first step is exactly the one described in the above skeleton, i.e., where A learns a_i and B learns b_i. In the second step of round i, the parties reconstruct a value x_i that describes the expected value of the game O_i. This extra step is not necessary for the correctness of the protocol, and hence, does not affect the security of the protocol (since any attack on the protocol not using x_i can also be applied to the protocol that gives x_i).

The basic two-party protocol of [24]. We now specify how the defense values are selected in the basic two-party protocol of [24] (parametrized by $\alpha \in [0,1]$), such that the common output bit is 1 with probability α. The basic idea is to sample $O(r^2)$ bits (i.e., elements from $\{-1,1\}$) i.i.d., such that the sum of all bits is positive with probability α. The prescribed output of the protocol is 1 if the sum of all bits is positive, and 0 otherwise. Towards revealing this output (gradually, in r rounds), let δ_i be the value of the game, conditioned on the value of the first $\sum_{k=r-i+1}^{r} k$ bits. Note that $\delta_0 = \alpha$ and that in each round i, the value of δ_i is computed conditioned on less and less *new* bits (i.e., bits that were not used to compute δ_{i-1}). The defense value given to each of the parties in round i is simply a sample from δ_i.

Slightly more formally, let $\varepsilon \in \left[-\frac{1}{2}, \frac{1}{2}\right]$ be such that the sum of $r(r+1)/2$ elements from $\{-1,1\}$ is positive with probability α, where each element is 1 with probability $1/2+\varepsilon$. Let x_i be the sum of $r-i+1$ elements from $\{-1,1\}$, where each element takes the value of 1 with probability $1/2+\varepsilon$. Let δ_i be the expected game value in round i, that is, δ_i is the probability that the sum of $\sum_{k=1}^{r-i} k$ elements from $\{-1,1\}$, is at least $\sum_{k=1}^{i} x_k$. The bits a_i and b_i are independently sampled according to δ_i, i.e., $a_i = 1$ (and $b_i = 1$) w.p. δ_i.

For some intuition on the security of the protocol, consider the case where party A, wishing to bias the output of party B, receives a defense value a_i before party B receives its defense value b_i. If A chooses to abort, then B is instructed to output a_{i-1}, which was sampled according to δ_{i-1}. Indeed, if A could see δ_i before deciding whether to abort or not, it could bias the output of B by $\Omega(1/\sqrt{r})$. The crux of the analysis is to show that this is not the case when A only receives a sample from δ_i. Towards this end, Haitner and Tsfadia [24] bound, on expectation, the gap between δ_{i-1} and $\widehat{\delta_{i-1}}$, defined to be the value of the game, conditioned on the value of the first $\sum_{k=r-i+1}^{r} k$ bits and on the value of a_i.

The three party protocol of [24]. The construction of [24] for three parties follows a very similar rationale to the above protocol. That is, in each round i every single party, as well as, every pair of parties obtain a defense value that should

behave as a sample from δ_i. A pair of parties cannot simply be given a single bit, since one of them may be corrupt. Rather, they should be given a two-party protocol similar to the above, with their defenses set with parameter $\alpha = \delta_i$. A problem arises here, since the simple application of the above idea would require giving the adversary information based on $\Omega(r^3)$ bits sampled according to the appropriate ε value. This would be devastating to the security of the protocol, as it would allow the adversary to reveal δ_i. To tackle this problem, [24] came up with a derandomized version of the above two-party protocol. They were then able to show that sending the shares for this protocol as the defense values for pairs of parties does not reveal too much about δ_i to the adversary. We next describe the derandomized two-party protocol of Haitner and Tsfadia [24].

The two-party derandomized protocol of [24]. We now specify how the defense values are selected in the derandomized version of the protocol of [24], such that the common output bit is 1 with probability α. Let $\varepsilon \in \left[-\frac{1}{2}, \frac{1}{2}\right]$ be such that the sum of $r(r+1)/2$ elements from $\{-1, 1\}$ is positive with probability α, where each element is 1 with probability $1/2 + \varepsilon$. For $j \in \{a, b\}$, let S^j be a set of size $r(r+1)$, over $\{-1, 1\}$, where each element takes the value of 1 with probability $1/2+\varepsilon$. Let x_i be the sum of $r-i+1$ elements from $\{-1, 1\}$, where each element takes the value of 1 with probability $1/2+\varepsilon$. Let δ_i^j be the expected game value in round i, according to the set S^j, that is, δ_i^j is the probability that the sum of the elements in a randomly chosen subset of S^j, of size $\sum_{k=1}^{r-i} k$, is at least $\sum_{k=1}^{i} x_k$. The bit a_i (respectively b_i) is sampled according to δ_i^a (respectively δ_i^b), i.e., $a_i = 1$ (respectively $b_i = 1$) with probability δ_i^a (respectively δ_i^b).

The security of the various constructions of [24] is proved via a series of bounds on weighted Binomial games. In Section 2, we recall these results, and in Section 3 we use them to prove the security of our construction.

1.3.3 Reducing Many-Party Coin-Tossing to Few-Party Coin-Tossing. Reducing multiparty coin-tossing protocols for the setting without an honest majority to 2-party protocols is quite straightforward. Indeed, the impossibility of Cleve [14] is generalized from the two-party setting to the many party setting via such a reduction. In this section, we show that sometimes the other direction is also possible.

The Protocol of Beimel et al. [8]. The protocol of Beimel, Omri, and Orlov [8] extends the results of [30] to the multiparty model, where fewer than 2/3 of the parties are corrupted. The bias of their protocol is proportional to $1/r$ and doubly exponential in the gap between the number of corrupted parties t and the number of honest parties h in the protocol ($m = h + t$). In particular, for a constant number of parties m, where fewer than $2m/3$ are corrupted, [8] present an r-round m-party coin-tossing protocol with an optimal bias of $O(1/r)$. Interestingly, their protocol has an $O(1/r)$-bias even when the number of parties m is non-constant, as long as the $t - h$ is constant. In the following description,

however, we present a simplified version of the protocol of [8], which requires t (rather than $t-h$) to be constant in order to achieve an $O(1/r)$-bias.

While not presented this way, the result of Beimel et al. [8] is achieved via a generic reduction to (a certain type of) two-party protocols. They use a few layers of secret sharing schemes to allow for each subset J of parties, containing an honest majority (i.e., $h \leq |J| < 2h$, hence if all the parties outside of J abort the execution, then there is an honest majority in J) to obtain a defense value, i.e., a bit d_i^J. For each round i and for each such J, the value of d_i^J is shared in an *inner* secret sharing scheme with threshold h-out-of-$|J|$. The idea is that the shares of this inner secret sharing scheme (of d_i^J) should be revealed to the parties of J at round i of the execution. Namely, each party in J should get one of the (inner scheme) shares of d_i^J in round i.

To make sure that the above shares are not revealed to any subset before round i, and at the same time, that the execution of the protocol proceeds, as long as, the set of remaining active parties does not contain an honest majority, the shares (of the inner scheme) for round i are shared in an *outer* secret sharing scheme with threshold $(t+1)$-out-of-m. As a result, the adversary can never learn anything about the shares of the i'th inner scheme without the help of honest parties. In addition, to halt the computation in round i, the adversary must instruct at least h parties to abort the computation.

Now, given a two-party protocol according to the above skeleton, and with the additional property that a_i and b_i are sampled from the same distribution D_i and that it is possible to sample many such samples, completing the reduction is done by selecting the defense values d_i^J from the distribution D_i.

If the following extra property holds, then the resulting many-party protocol would be α-fair as long as $t < 2m/3$. The extra property that we need to require is that if the adversary in the 2-party protocol is given 2^{2^m} defense values, sampled from D_i (and the honest party gets a single one), it will not be able to bias the 2-party protocol by more than α.[2]

1.3.4 Applying the Reduction of Beimel et al. to the Protocols of Haitner and Tsfadia.

In this work, we use secret sharing schemes, in a manner similar to [8], to reduce an m-party coin-tossing with $t < 3m/4$ malicious to the 3-party construction of [24]. We do so in two steps. First, we apply the above (simplified version of the) reduction of [8] to the (derandomized) two-party protocol of [24] to obtain an *auxiliary* $\hat{m}$-party coin-tossing protocol, tolerating $\hat{t} < 2\hat{m}/3$ corruptions. Then, we use the auxiliary protocol, as a building block in the construction of the *final* m-party protocol that tolerates $t < 3m/4$ corruptions. More specifically, the auxiliary protocol, parametrized by some $\varepsilon \in [0,1]$, is used as defense values for subsets of parties for the case that at least $m/4$ corrupted parties abort the execution of the final protocol.

[2] Beimel et al. [8] use a slightly more involved technique to distribute defense values to the different subsets of parties, allowing several subsets to be assigned the same output bit, while maintaining the guarantee that the adversary cannot bias the output of the honest parties without guessing the value of the special round i^*.

We next give an overview of both constructions. In Section 1.4, we exemplify the constructions for the case that $m = 7$ and $t = 5$; in Section 1.4.1, we instantiate the auxiliary protocol for the case of five parties with up to three corruptions, and in Section 1.4.2, we use this construction to instantiate the final protocol for the case of seven parties with up to five corruptions. In the following, let $\hat{h} = \hat{m} - \hat{t}$ and $h = m - t$ be lowerbounds on the number of honest parties in the respective protocols. In our discussion the auxiliary protocol will be used with $\hat{m}$ being the number of active parties remaining after some corrupted parties have prematurely aborted the execution of the final m-party protocol. Specifically, we will have $\hat{h} = h$, since honest parties never prematurely abort the computation.

Both the basic and the final protocols use two layers of (threshold) secret sharing schemes. For each round i and for each *protected* subset of parties J (we specify below which subsets are called protected for each construction), the defense value for the set J in round i is d_i^J. This defense value is shared among the parties of J in an appropriate secret sharing scheme (actual parameters for each construction are specified below). This is called the *inner* secret sharing scheme. For each round i, all the shares of all parties in the inner secret sharing schemes for round i are shared in an $(\tilde{t}+1)$-out-of-$\tilde{m}$ threshold secret sharing scheme, where $\tilde{m}$ and $\tilde{t}$ are the number of parties and the bound on the number of corruptions in the respective construction. This is called the *outer* secret sharing scheme.

The idea behind the outer secret sharing scheme is to provide two guarantees. First, the adversary is never able to reconstruct the secrets without the participation of honest parties (which will only participate in the appropriate round). Second, the adversary is only able to prevent the reconstruction of the secret of the outer scheme (for round i) by instructing at least $\tilde{h} = \tilde{m} - \tilde{t}$ corrupted parties to abort before completing the reconstruction. Hence, the protocol proceeds normally as long as more than $\tilde{t}$ parties are active. We stress that the adversary is indeed able to instruct $\tilde{h}$ parties to abort in the process of reconstruction of the secret of the outer secret sharing scheme, hence, seeing all the shares of corrupted parties for round i, while not allowing honest parties to see their shares of the inner scheme. Furthermore, since the adversary is rushing, it can actually decide whether to do so or not – after seeing the shares of all honest parties.

In addition to the above, assume that $\tilde{t} < \frac{b\tilde{m}}{b+1}$ for some natural $b > 1$, and assume that at least $\tilde{h}$ corrupted parties aborted (which is the case if the secret of the outer scheme cannot be reconstructed). Let J be the set of the remaining parties and let t_J be the number of corrupted parties in J. Since the number of honest parties in J remains the same as before, i.e., at least $h > \frac{\tilde{m}}{b+1}$, it follows that $t_J < |J| - \frac{\tilde{m}}{b+1}$. By assumption $|J| \leq \tilde{t} < \frac{b\tilde{m}}{b+1}$, it follows that $t_J < |J| - \frac{\tilde{m}}{b+1} < \frac{(b-1)\cdot|J|}{b}$. Thus, if h parties abort the execution of the final construction, then less than $2/3$ of the remaining parties are corrupted, and if $\hat{h}$ parties abort the execution of the auxiliary construction, then most of the remaining parties are honest.

We now explain what protected subsets are and how the parameters for the inner secret sharing schemes are chosen for each of the two constructions. We begin with the final construction. Protected subsets of parties are subsets J that are assigned a defense value d_i^J in each round i. These should include all subsets that are liable to become the set of active parties, after a premature abort by at least h parties. Since the number of aborting (corrupted) parties may be anything between h and t, we should let protected subsets be all subsets of parties J, such that $h \leq |J| \leq t$.[3]

To determine the parameters for the inner secret sharing scheme, consider the case that $a \geq h$ corrupted parties have aborted in round i, hence the set of active parties J is of size $m - a$. Let t_J be the number of corrupted parties in J, then $t_J \leq t - a$. Therefore, using a $(t - m + |J| + 1)$-out-of-$|J|$, we require at least $t - a + 1 = t - m + |J| + 1$ parties of J for the reconstruction of d_i^J. This ensures that the adversary was never able to reconstruct d_{i-1}^J (which is the defense value that the parties in J will use). Very similar reasoning are used for the auxiliary construction, where a subset of parties is protected if it of any size between $\hat{h}$ and $2\hat{h} - 1$, and the threshold of the inner secret sharing scheme is set to $\hat{h}$-out-of-$|J|$.

It is left to specify what are the defense values d_i^J, which are the secrets that are shared in the inner secret sharing schemes. Roughly speaking these values are selected in the auxiliary and in the final constructions in a very similar manner to that of the derandomized two-party and the three-party protocols of [24] (respectively). In a bit more detail, in these protocols, there is a value δ_i representing the expected value of the game, and the defense values for all protected subsets describe a way to reveal a sample a bit according to δ_i.

In the final protocol, a defense value is an instantiation of the auxiliary protocol, such that the output bit is 1 with probability δ_i. To be more precise, d_i^J is the set of shares in the outer secret sharing of the instantiation of the auxiliary protocol to be executed by the parties of J, in case all other parties abort the computation. The exact same information can also be encapsulated into a set of $O(r^2)$ elements from $\{-1, 1\}$ taking the value with probability $1/2 + \varepsilon$, where $\varepsilon = \varepsilon(\delta_i) \in \left[-\frac{1}{2}, \frac{1}{2}\right]$ is such that the sum of $r(r+1)/2$ elements from $\{-1, 1\}$ is positive with probability δ_i, whenever each element is 1 with probability $1/2 + \varepsilon$. Indeed, this fact will allow us to use the vector game lemma of [24] (see Lemma 2) to bound the bias that the adversary can inflict by seeing the defense values of all corrupted protected sets. The proof of security of the final protocol is obtained by combining the above bound with a bound on the bias of the auxiliary protocol.

We now specify how the defense values are selected in the auxiliary protocol. Let J be a protected subset of parties, the parties of J jointly hold a set S^J of size $r(r+1)$, over $\{-1, 1\}$, where each element takes the value of 1 with probability $1/2 + \varepsilon$. Recall that x_i be the sum of $r - i + 1$ elements from $\{-1, 1\}$, where each

[3] Actually, in our construction, we only call subsets J, such that $2h - 1 \leq |J| \leq t$. This suffices, since if a smaller subset of active parties is left, it can use the defense value of its lexicographically first superset of size $2h - 1$.

element takes the value of 1 with probability $1/2+\varepsilon$. Let δ_i^J be the expected game value in round i, according to the set S^J, that is, δ_i^J is the probability that the sum of the elements in a randomly chosen subset of S^J, of size $\sum_{k=1}^{r-i} k$, is at least $\sum_{k=1}^{i} x_k$. The bit b_i^J is sampled according to δ_i^J, i.e., $b_i^J = 1$ with probability δ_i^J. To prove the security of this protocol, we introduce an extended version of the Hypergeometric game (Lemma 3), presented in [24]. More specifically, we show that even when the adversary sees a (constant) number of independent samples, each from a different set, it cannot bias the output by much.

1.4 A Warm-Up Construction – A Seven-Party Protocol Tolerating up to Five Corrupted Parties

Following the overview of our constructions, given in Section 1.3.4, in this section, we show how to instantiate our final construction for the case of 7 parties, where at most 5 are corrupted. In Section 1.4.1, we instantiate the auxiliary protocol for 5 parties with at most 3 corruptions, and in Section 1.4.2 we use it to instantiate the final protocol for 7 parties with at most 5 corruptions. In the following, let $\varepsilon \in \left[-\frac{1}{2}, \frac{1}{2}\right]$, and for $i \in \{0, \ldots, r\}$ let $s_i = \sum_{k=1}^{r-i} k$.

1.4.1 A Five-Party Protocol Tolerating up to Three Corrupted Parties.

We now describe the algorithm $HG(\varepsilon, 5, 3)$, generating shares for 5 parties with 3 corrupted parties. This is a specific instantiation of the more general functionality described in Algorithm 5. Let $\mathcal{B}in_{n,\varepsilon}$ denote the binomial distribution over $\{-1, 1\}$ (i.e., a sum of n samples from $\{-1, 1\}$, each taking the value of 1 with probability $\frac{1}{2} + \varepsilon$).

Selecting defenses:

1. For every $J \subset [5]$ of size 3, let S^J be a set with $2s_0$ elements from $\{-1, 1\}$, each taking the value of 1 with probability $\frac{1}{2} + \varepsilon$.
2. For every $i \in [r]$ let $\hat{x}_i \leftarrow \mathcal{B}in_{r-i+1,\varepsilon}$.
3. For every $i \in \{0, \ldots, r\}$ and every $J \subseteq [5]$ of size 3:
   (a) Let A_i^J be a random subset of S^J of size s_i.
   (b) Let $\hat{d}_i^J$ be 1 if $\sum_{k=1}^{i} \hat{x}_k + \sum_{a \in A_i^J} a \geq 0$, and 0 otherwise.

Sharing the values:

- For every $i \in \{0, 1 \ldots, r\}$, $J \subset [5]$ of size 3, and $j \in J$, let $d_i^J[j]$ be the share of party P_j of the secret d_i^J, in a 2-out-of-3 secret sharing.
- For every $i \in [r]$, $J \subset [5]$ of size 3, and for every $j' \in J$, let $d_i^J[j', j]$ be the share of party P_j of the secret $d_i^J[j']$, in a 4-out-of-5 secret sharing, such that party $P_{j'}$ is required in order to recover $d_i^J[j']$ (See Construction 4).

Interaction rounds. The interaction of the parties proceeds in r rounds. In round $i \in [r]$, party P_j broadcasts $d_i^J[j', j]$, for every $J \subset [5]$ of size 3, and for every $j' \in J$. If a single party aborts the execution, then the remaining 4 parties can continue with the protocol. If two or three parties abort the execution, then the

remaining parties reconstruct $d^J_{i'}$, where J is lexicographically first set of size 3, which contains all the indices of the active parties, and i' is the maximum i for which the parties have enough shares to reconstruct. The honest parties output that bit.

If after r rounds, there are at least 4 active parties, then the parties reconstruct the last joint defense for the lexicographically first subset of them, and the honest parties output that bit.

Security. By the properties of the two layers of secret sharing, in each round the adversary learns a constant number of defense values, which are sampled according to the appropriate Hypergeometric distribution. Roughly speaking, the security of the above protocol is reduced to an extended version of the Hypergeometric game considered by [24], with a constant number of samples. The proof of security of the general construction, as well as, the froof of the bound for the extended Hypergeometric game are given in the full version of the paper [1].

1.4.2 The Seven-Party Protocol.

We are now ready to describe our 7 party protocol. We first describe the share generator. Given $x_1 \dots x_i$, for some $i \in [r]$ we let $\delta_i(x_1 \dots x_i)$ be the probability that then sum of s_i uniform $\{-1, 1\}$ bits is at least $-\sum_{k=1}^{i} x_k$. We call δ_i the expected outcome of the protocol in round i. In the following we let $\mathcal{B}in_n := \mathcal{B}in_{n,0}$.

Selecting defenses:

1. For every $i \in [r]$, let $x_i \leftarrow \mathcal{B}in_{r-i+1}$.
2. Let $\varepsilon_i \in \left[-\frac{1}{2}, \frac{1}{2}\right]$ be such that, the expected outcome of an honest execution with parameter $\varepsilon = \varepsilon_i$ of the 5-party protocol from Section 1.4.1 is $\delta_i(x_1 \dots x_i)$.
3. For every $J \subset [7]$, such that $4 \leq |J| \leq 5$, let $d^J_i \leftarrow HG(\varepsilon_i, |J|, |J| - 2)$.
4. For every $J \subset [7]$, such that $2 \leq |J| \leq 3$, let d^J_i be a bit, sampled with probability $\delta_i(x_1 \dots x_i)$.

Sharing the values:

- For every $i \in [r]$ and $J \subset [7]$, such that $4 \leq |J| \leq 5$, let $d^J_i[j]$ be the share of party P_j of the secret d^J_i, in a $(|J|-1)$-out-of-$|J|$ secret sharing.
- For every $i \in [r]$, $J \subset [7]$, such that $4 \leq |J| \leq 5$, and for every $j' \in J$, let $d^J_i[j', j]$ be the share of party P_j of the secret $d^J_i[j']$, in a 6-out-of-7 secret sharing, such that party $P_{j'}$ is required in order to recover $d^J_i[j']$ (See Construction 4).
- For every $i \in [r]$ and $J \subset [7]$, such that $2 \leq |J| \leq 3$, let $d^J_i[j]$ be the share of party P_j of the secret d^J_i, in a 2-out-of-$|J|$ secret sharing.

Interaction rounds. The interaction of the parties proceeds in r rounds. In round $i \in [r]$ party P_j broadcasts $d^J_i[j', j]$, for every $J \subset [7]$, such that $3 \leq |J| \leq 5$, and for every $j' \in J$.

If a single party aborts the execution, then the remaining 6 parties can continue with the protocol (they can do so by the properties of the 6-out-of-7 secret

sharing scheme). If more parties abort the execution, then the remaining active parties reconstruct $d_{i'}^J$, where J is the lexicographic first set containing all their indices, and i' is the maximum i for which the parties have enough shares to reconstruct. If more than three parties remain, then they execute the five party protocol from Section 1.4.1. Otherwise, there is an honest majority, and hence, the remaining parties reconstruct $d_{i'}^J$, which is a bit.

If after r rounds, there are at least 5 active parties, then each pair reconstruct its last common defense (Note that either all of these defenses are equal to 1 or all of them are equal to 0).

Security. In each round $i \in [r]$, the adversary learns an $O\left(r^2\right)$ bits sampled according to ε_i. If only one party aborts the execution, then the remaining parties can still continue, as the secret sharing is a 6-out-of-7. Hence the adversary must instruct at least two parties to abort. In case at least two parties abort at round i, the remaining active parties can reconstruct the defense from the round $i-1$. They then, execute the protocol described in Section 1.4.1. As this is the Vector game considered by [24], the adversary does not gain much advantage from aborting after seeing the above $O\left(r^2\right)$ bits samples (assuming that the remaining parties run the defense protocol honestly). Of course, we cannot assume that they do, however, combining the above with the security of the 5-party protocol, we get that in total, the adversary's gain remains small.

1.5 Organization

In Section 2, we provide some notations and definitions that we use in this work, and recall some bounds ononline Binomial games from [24]. In Section 3 we present our main construction and provide a proof for Theorem 1.

2 Preliminaries

2.1 Notation

We use calligraphic letters to denote sets, uppercase for random variables, and lowercase for values. All logarithms considered here are in base two. For $n \in \mathbb{N}$, let $[n] = \{1, 2 \ldots n\}$. Given a random variable (or a distribution) X, we write $x \leftarrow X$ to indicate that x is selected according to X. The support of a distribution D over a finite set S, denoted $\mathrm{Supp}(D)$, is defined as $\{s \in S \mid D(s) > 0\}$. For a random variable X and a natural number n we let $X^n = \left(X^{(1)}, X^{(2)}, \ldots, X^{(n)}\right)$, where the $X^{(i)}$'s are i.i.d. copies of X.

Let $n \in \mathbb{N}$ and $\varepsilon \in \left[-\frac{1}{2}, \frac{1}{2}\right]$. Let $\mathcal{B}er(\varepsilon)$ be the Bernoulli distribution over $\{-1, 1\}$, taking 1 with probability $\frac{1}{2}+\varepsilon$. Define the Binomial distribution $\mathcal{B}in_{n,\varepsilon}$, by $\mathcal{B}in_{n,\varepsilon}(k) = \Pr\left[\sum_{i=1}^{n} x_i = k\right]$ where x_i are i.i.d according to $\mathcal{B}er(\varepsilon)$. Let $\widehat{\mathcal{B}in}_{n,\varepsilon}(k) = \Pr_{x \leftarrow \mathcal{B}in_{n,\varepsilon}}[x \geq k] = \sum_{t \geq k} \mathcal{B}in_{n,\varepsilon}(t)$. For $\varepsilon = 0$ we will simply write $\mathcal{B}in_n$ and $\widehat{\mathcal{B}in}_n$.

Define the Hypergeometric distribution $\mathcal{HG}_{n,w,m}$, by $\mathcal{HG}_{n,w,m}(k) = \Pr_{S\subseteq\mathcal{S},|S|=m}\left[\sum_{s\in S} s = k\right]$, where S is chosen uniformly, $\mathcal{S}$ is a set of size n, whose members are from $\{-1,1\}$, and it holds that $\sum_{s\in\mathcal{S}} s = w$. Let $\widehat{\mathcal{HG}}_{n,w,m}(k) = \Pr_{x\leftarrow\mathcal{HG}_{n,w,m}}[x \geq k] = \sum_{t\geq k} \mathcal{HG}_{n,w,m}(t)$. For $i \in \{0,1,\ldots n\}$ let $s_i(n) = \sum_{k=1}^{n-i} k = \frac{(n-i+1)(n-i)}{2}$. When n is clear from the context we write s_i. For a set S we let $w(S) = \sum_{s\in S} s$.

We make use of the following facts.

Fact 2 (Hoeffding's inequality for $\{-1,1\}$) *Let $n,t \in \mathbb{N}$ and let $\varepsilon \in \left[-\frac{1}{2},\frac{1}{2}\right]$. Then*

$$\Pr_{x\leftarrow\mathcal{B}in_{n,\varepsilon}}[|x-2\varepsilon n| \geq t] \leq 2e^{-\frac{t^2}{2n}}.$$

Fact 3 (Hoeffding's inequality for the hypergeometric distribution) *Let $m \leq n \in \mathbb{N}$ and let $w \in \mathbb{Z}$ satisfying $|w| \leq n$. Then*

$$\Pr_{x\leftarrow\mathcal{HG}_{n,w,m}}[|x-\mu| \geq t] \leq e^{-\frac{t^2}{2m}},$$

where $\mu = \mathrm{E}_{x\leftarrow\mathcal{HG}_{n,w,m}}[x] = \frac{mw}{n}$

2.2 Coin-Tossing Protocols

A multiparty coin-tossing protocol with m parties is defined using m probabilistic polynomial-time Turing machines $p_1,\ldots,p_m$ having the security parameter 1^n as their only input. The coin-tossing computation proceeds in rounds, in each round, the parties broadcast and receive messages on a broadcast channel. The number of rounds in the protocol is typically expressed as some polynomially-bounded function r in the security parameter. At the end of protocol, the (honest) parties should hold a common bit w. We denote by $\mathrm{CoinToss}_\varepsilon()$ the ideal functionality that gives the honest parties the same bit w, distributed according to ε, that is, $\Pr[w=1] = 1/2+\varepsilon$ and $\Pr[w=0] = 1/2-\varepsilon$. We let $\mathrm{CoinToss}()$ be $\mathrm{CoinToss}_0()$.

In this work we consider a malicious static computationally-bounded adversary, i.e., a non-uniform that runs in a polynomial-time. The adversary is allowed to corrupt some subset of the parties. That is, before the beginning of the protocol, the adversary corrupts a subset of the parties that may deviate arbitrarily from the protocol, and thereafter the adversary sees the messages sent to the corrupt parties and controls the messages sent by the corrupted parties. Still, for the most of the technical discussion of the paper, we only discuss fail-stop adversaries. A fail-stop adversary acts completely honestly (i.e., as required by the prescribed protocol), with the only difference that it can abort the computation at any point in the execution of the protocol. We, then, use standard techniques ([19, 8]) to turn a coin-tossing protocol in the fail-stop model into a coin-tossing protocol (with the same fairness and round-complexity) in the malicious model. The honest parties follow the instructions of the protocol.

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

2.3 Security Definitions for Multiparty Protocols

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 at hand. This ideal model involves a trusted party whose functionality captures the security requirements of the task. Finally, we show that the real-world protocol "emulates" the ideal-world protocol: For any real-life adversary $\mathcal{A}$ there should exist an ideal-model adversary $\mathcal{S}$ (also 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 the coin-tossing protocol, the parties do not have inputs. Thus, to simplify the definitions, we define secure computation without inputs (except for the security parameters).

The Real Model. Let Π be an m-party protocol computing $\mathcal{F}$. Let $\mathcal{A}$ be a non-uniform probabilistic polynomial time adversary with auxiliary input aux, corrupting a subset $\mathcal{C}$ of the parties. Let $REAL_{\Pi,\mathcal{A}(\text{aux})}(1^n)$ be the random variable consisting of the view of the adversary (i.e., its random input and the messages it got) and the output of the honest parties, following an execution of Π, where each party p_j begins by holding the input 1^n.

The Ideal Model. The basic ideal model we consider is a model without abort. Specifically, there are parties $\{p_1, \ldots, p_m\}$, and an adversary $\mathcal{S}$ who has corrupted a subset I of them. An ideal execution for the computing $\mathcal{F}$ proceeds as follows:

Inputs: Party p_j holds a security parameter 1^n. The adversary $\mathcal{S}$ has some auxiliary input aux.

Trusted party sends outputs: The trusted party computes $\mathcal{F}(1^n)$ with uniformly random coins and sends the appropriate outputs to the parties.

Outputs: The honest parties output whatever they received from the trusted party, the corrupted parties output nothing, and $\mathcal{S}$ outputs an arbitrary probabilistic polynomial-time computable function of its view.

Let $IDEAL_{\mathcal{F},\mathcal{S}(\text{aux})}(1^n)$ 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.

In this work we consider a few formulations of the ideal-world, and consider composition of a few protocols, all being executed in the same real-world, however, each secure with respect to a different ideal-world. We prove the security of the resulting protocol, using the hybrid model techniques of Canetti [13].

2.3.1 $1/p$-Indistinguishability and $1/p$-Secure Computation. As explained in the introduction, the ideal functionality CoinToss() cannot be implemented when there is no honest majority. We use $1/p$-secure computation, defined by [20, 27], to capture the divergence from the ideal world. This notion applies to general secure computation. We start with some notation.

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* $X = \{X_{a,n}\}_{a\in\{0,1\}^*,n\in\mathbb{N}}$ is an infinite sequence of random variables indexed by $a \in \{0,1\}^*$ and $n \in \mathbb{N}$.

Definition 1 (Statistical Distance and $1/p$-indistinguishability). *We define the* statistical distance *between two random variables A and B as the function*

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

For a function $p(n)$, two distribution ensembles $X = \{X_{a,n}\}_{a\in\{0,1\}^*,n\in\mathbb{N}}$ and $Y = \{Y_{a,n}\}_{a\in\{0,1\}^*,n\in\mathbb{N}}$ 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 $a \in \{0,1\}^*$,*

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

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

We next define the notion of $1/p$-secure computation [20, 27, 7]. The definition uses the standard real/ideal paradigm [18, 13], 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 (perfect $1/p$-secure computation). *An m-party protocol Π is said to* perfectly $(t,1/p)$*-secure compute a functionality $\mathcal{F}$ if for every non-uniform adversary $\mathcal{A}$ in the real model, corrupting up to t of the parties, there exists a polynomial-time adversary $\mathcal{S}$ in the ideal model, corrupting the same parties as $\mathcal{A}$, such that for every $n \in \mathbb{N}$ and for every* aux $\in \{0,1\}^*$

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

Definition 3 ($1/p$-secure computation [20, 27, 7]). *Let $p = p(n)$ be a function. An m-party protocol Π is said to $(t,1/p)$-securely compute a functionality $\mathcal{F}$ if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ in the real model, corrupting up to t of the parties, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model, corrupting the same parties*

as $\mathcal{A}$, such that the following two distribution ensembles are computationally $1/p(n)$-indistinguishable

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

We next define the notion of *secure computation* and notion of *bias of a coin-tossing protocol* by using the previous definition.

Definition 4 (secure computation). *An m-party protocol Π t-*securely computes *a functionality $\mathcal{F}$, if for every $c \in N$, the protocol Π is $(t, 1/n^c)$-securely compute the functionality $\mathcal{F}$.*

Definition 5 (ε-coin-toss). *We say that a protocol is a ε-coin-toss protocol with bias $1/p$, tolerating up to t corruptions, if it is a $(t, 1/p)$-secure protocol for the functionality* $\text{CoinToss}_\varepsilon()$.

Definition 6 (coin tossing). *We say that a protocol is a coin-tossing protocol with bias $1/p$, tolerating up to t corruptions, if it is a $(t, 1/p)$-secure protocol for the functionality* $\text{CoinToss}()$.

2.4 Security with Identifiable Abort

We use here a variant of secure computation with abort, where upon abort, at least one cheating party is identified to all honest parties. This definition was first formally stated by Aumann and Lindell [5], and was also considered in [8, 7, 26], (in the first two, it was called security with abort and cheat detection).

Roughly speaking, our definition requires that one of two events is possible: If at least one party deviates from the prescribed protocol, then the adversary obtains the outputs of these parties (but nothing else), and all honest parties are notified by the protocol that these parties have aborted. Otherwise, the protocol terminates normally, and all parties receive their outputs. Again, we consider the restricted case where parties hold no private inputs. The formal definition is omitted for lack of space, and will appear in the full version of the paper [1].

2.5 Cryptographic Tools

We next informally describe two cryptographic tools that we use in our protocols.

Signature Schemes. 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 $(\text{Gen}, \text{Sign}, \text{Ver})$ containing the key generation algorithm Gen, which gets as input a security parameter 1^n and 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.

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 parties of size α can reconstruct the secret, while any smaller set knows nothing about the secret. In this paper, we use Shamir's α-out-of-m secret-sharing scheme [33]. In this scheme, the shares of any $\alpha - 1$ parties are uniformly distributed and independent of the secret. Furthermore, given at most such $\alpha - 1$ shares and a secret s, one can *efficiently* complete them to m shares of the secret s. Using this scheme, [8] presented a way to construct a secret sharing scheme *with respect to a certain party*. We use that in our construction as well.

Construction 4 *Let s be some 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 $\left(s^{(1)}, s^{(2)}\right)$ 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. *Generate shares $\left(\lambda^{(1)}, \ldots, \lambda^{(j-1)}, \lambda^{(j+1)}, \ldots, \lambda^{(m)}\right)$ 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, the secret s is shared among the parties in P in a secret-sharing scheme such that any set of size at least α that contains p_j can reconstruct the secret. In addition, similarly to the Shamir secret-sharing scheme, the following property holds: for any set of $\beta < \alpha$ parties (regardless if the set contains p_j), 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 and *efficiently* select uniformly at random one vector of shares competing the β shares to m shares of the secret s.

2.6 Claims and Definitions from [24]

The following definitions and propositions are taken verbatim from [24] and they will serve us as well. Given a partial view of a fail-stop adversary, we are interested in the expected outcome of the parties, conditioned on this view and the adversary making no further aborts.

Definition 7 (view value). *Let π be a protocol in which the honest parties always output the same bit value. For a partial view v of the parties in a fail-stop execution of π, let $C_\pi(v)$ denote the parties full view in an honest execution of π conditioned on v (i.e. all parties that do not abort in v act honestly in $C_\pi(v)$). Let $\Delta_\pi(v) = \mathrm{E}_{v' \leftarrow C_\pi(v)}[out(v')]$, where $out(v')$ is the common output of the non-aborting parties in v'.*

A protocol is unbiased, if no fail-stop adversary can bias the common output of the honest parties by too much.

Definition 8 ((t,α)-unbiased protocol). *Let π be an m-party, r-round protocol, in which the honest parties always output the same bit value. We say that π is (t,α)-unbiased, if the following holds for every fail-stop adversary $\mathcal{A}$ controlling the parties indexed by a subset $\mathcal{C} \subset [m]$ of size at most t. Let V be $\mathcal{A}$'s view in a random execution of π, and let I_j be the index of the j'th round in which $\mathcal{A}$ sent an abort message (set to $r+1$ if no abort occurred). Let V_i be the prefix of V at the end of the i'th round, letting V_0 be the view consisting of only the random coins of $\mathcal{A}$, and let V_i^- be the prefix of V_i with the i'th round abort message (if any) removed. Then,*

$$\mathop{\mathrm{E}}_{V}\left[\left|\sum_{j\in|\mathcal{C}|}\left(\Delta(V_{I_j})-\Delta(V_{I_j}^-)\right)\right|\right]\leq\alpha$$

where $\Delta=\Delta_\pi$ according to Definition 7.

The following is an alternative characterization of fair coin-tossing protocols (against fail-stop adversaries).

Lemma 1 ([24, Lemma 2.18]). *Let $n\in\mathbb{N}$ be a security parameter and let π be a (t,α)-unbiased coin-tossing protocol with $\alpha(n)\leq\frac{1}{2}-\frac{1}{p(n)}$, for some polynomial p. Then π is a $(t,\alpha(n)+neg(n))$-secure coin tossing protocol against fail-stop adversaries.*

The following lemmata and propositions assume that the protocol is of a specific form. More concretely, let $\varepsilon\in\left[-\frac{1}{2},\frac{1}{2}\right]$, f be a randomized function (that may depend on ε), and let $\pi_{\varepsilon,f}$ be an r-round m-party coin-tossing protocol, such that, before any interaction takes place, every party learns D_0, which is sampled according to the current game value, and for every round $i\in[r]$, every party first learns a defense $D_i=f(i,Y_i)$, and then the coin X_i, where $X_i\leftarrow\mathcal{B}in_{r-i+1,\varepsilon}$, $Y_i=\sum_{k=1}^{i}X_k$. We let $V_{\pi_{\varepsilon,f}}$ denote the adversary's view in a random execution of $\pi_{\varepsilon,f}$. We further assume that adversary never aborts after seeing X_i.

Lemma 2 (Vector Game [24, Lemma 4.5]). *Let $c\in\mathbb{N}$ and let $r\in\mathbb{N}$ be the number of rounds. Let $f:[r]\times\mathbb{Z}\to\{-1,1\}^{c\cdot r^2}$ be a randomized function that on input (i,y) outputs $c\cdot r^2$ elements from $\{-1,1\}$, each takes the value of 1 with probability $\mathcal{B}er(\varepsilon)$, where $\varepsilon\in\left[-\frac{1}{2},\frac{1}{2}\right]$ satisfies $\widehat{\mathcal{B}in}_{s_0,\varepsilon}(0)=\widehat{\mathcal{B}in}_{s_i}(-y)$. Then:*

$$\mathop{\mathrm{E}}_{V_{\pi_{0,f}}}\left[\left|\Delta\left(V_{\pi_{0,f}}\right)-\Delta\left(V_{\pi_{0,f}}^-\right)\right|\right]=O\left(\frac{\log^3 r}{r}\right).$$

Lemma 3 (Hypergeometric Game [24, Lemma 4.4]). *Let $w\in\mathbb{Z}$, $\varepsilon\in\left[-\frac{1}{2},\frac{1}{2}\right]$ and let $r\in\mathbb{N}$ be the number of rounds. Let $f:[r]\times\mathbb{Z}\to\{0,1\}$ be a randomized function that on input (i,y) outputs 1 with probability $\widehat{\mathcal{HG}}_{2s_0,w,s_i}(-y)$*

and 0 otherwise. Assuming that $|w| \leq c \cdot \sqrt{\log r \cdot s_0}$*, for some constant* c*, then:*

$$\mathop{\mathrm{E}}_{V_{\pi_{\varepsilon,f}}}\left[\left|\Delta\left(V_{\pi_{\varepsilon,f}}\right) - \Delta\left(V^{-}_{\pi_{\varepsilon,f}}\right)\right|\right] = O\left(\frac{\log^3 r}{r}\right).$$

Lemma 4 (Ratio Lemma [24, Lemma 4.10]). *Let* $r \in \mathbb{N}$ *be the number of rounds, and let* $\varepsilon \in \left[-\frac{1}{2}, \frac{1}{2}\right]$*. In the following we let* $Y_0 = 0$*. Let*

$$\mathcal{X}_i := \left\{x \in \mathrm{Supp}(X_i) : |x| \leq 4\sqrt{\log r \cdot (r-i+1)}\right\}$$

and

$$\mathcal{Y}_i := \left\{y' \in \mathrm{Supp}(Y_{i-1}) : |y' + 2\varepsilon \cdot s_{i-1}| \leq 4\sqrt{\log r \cdot s_{i-1}}\right\}.$$

Assume $|\varepsilon| \leq 2\sqrt{\frac{\log r}{s_0}}$ *and that for every* $i \in [r - \lfloor \log^{2.5} r \rfloor]$ *and* $y \in \mathcal{Y}_i$*, there exists a set* $\mathcal{D}_{i,y}$ *such that for every* $x \in \mathcal{X}_i$*, and every* $d \in \mathcal{D}_{i,y} \cap \mathrm{Supp}(f(i, y + X_i) \mid Y_{i-1} = y, X_i \in \mathcal{X}_i)$*, it holds that:*

$$\Pr[f(i, y + X_i) \notin \mathcal{D}_{i,y} \mid Y_{i-1} = y] \leq \frac{1}{r^2}$$

and

$$\left|1 - \frac{\Pr[f(i, y + X_i) = d \mid Y_{i-1} = y \wedge X_i = x]}{\Pr[f(i, y + X_i) = d \mid Y_{i-1} = y \wedge X_i \in \mathcal{X}_i]}\right| \leq c \cdot \sqrt{\frac{\log r}{r-i}} \cdot \left(1 + \frac{|x|}{\sqrt{r-i+1}}\right),$$

for some constant c*. Then:*

$$\mathop{\mathrm{E}}_{V_{\pi_{\varepsilon,f}}}\left[\left|\Delta\left(V_{\pi_{\varepsilon,f}}\right) - \Delta\left(V^{-}_{\pi_{\varepsilon,f}}\right)\right|\right] = O\left(\frac{\log^3 r}{r}\right).$$

Proposition 1 ([24, Proposition 4.6]). *For every randomized functions* f, g*, and for every* $\varepsilon \in \left[-\frac{1}{2}, \frac{1}{2}\right]$*, it holds that*

$$\mathop{\mathrm{E}}_{V_{\pi_{\varepsilon,g\circ f}}}\left[\left|\Delta\left(V_{\pi_{\varepsilon,g\circ f}}\right) - \Delta\left(V^{-}_{\pi_{\varepsilon,g\circ f}}\right)\right|\right] \leq \mathop{\mathrm{E}}_{V_{\pi_{\varepsilon,f}}}\left[\left|\Delta\left(V_{\pi_{\varepsilon,f}}\right) - \Delta\left(V^{-}_{\pi_{\varepsilon,f}}\right)\right|\right]$$

Proposition 2 ([24, Proposition 4.7]). *Let* $\varepsilon \in \left[-\frac{1}{2}, \frac{1}{2}\right]$ *and* f *be some randomized function. If* $\Pr[Y_r \geq 0] \notin \left[\frac{1}{r^2}, 1 - \frac{1}{r^2}\right]$*, where* $r \in \mathbb{N}$ *is the number of rounds, then*

$$\mathop{\mathrm{E}}_{V_{\pi_{\varepsilon,f}}}\left[\left|\Delta\left(V_{\pi_{\varepsilon,f}}\right) - \Delta\left(V^{-}_{\pi_{\varepsilon,f}}\right)\right|\right] \leq \frac{2}{r}.$$

2.7 An Extension of the Hypergeometric Game

In this section we introduce an extended version of the Hypergeometric game (Lemma 3), presented in [24]. More specifically, we let the adversary see a constant number of independent samples, each from a different set. Furthermore, we augment the view of the adversary with all of these sets.

Lemma 5. *Let $\xi \in \mathbb{N}$ be some constant, let $\boldsymbol{w} = (w_1 \ldots, w_\xi) \in \mathbb{Z}^\xi$, let $\varepsilon \in \left[-\frac{1}{2}, \frac{1}{2}\right]$, and let $r \in \mathbb{N}$ be the number of rounds. For $k \in [\xi]$, let $h_k : [r] \times \mathbb{Z} \to \{0,1\}$ be a randomized function that on input (i, y) outputs 1 with probability $\widehat{\mathcal{HG}}_{2s_0,w_k,s_i}(-y)$ and 0 otherwise. Assuming that for every $k \in [\xi]$, it holds that $|w_k| \leq c\sqrt{\log r \cdot s_0}$, for some constant c, then:*

$$\mathop{\mathrm{E}}_{V_{\pi_{\varepsilon,h}}}\left[\left|\Delta\left(V_{\pi_{\varepsilon,h}}\right) - \Delta\left(V^{-}_{\pi_{\varepsilon,h}}\right)\right|\right] = O\left(2^{\xi} \cdot \frac{\log^3 r}{r}\right),$$

where $h(i, y) = (h_1(i, y), \ldots, h_\xi(i, y))$.

The proof of Lemma 5 is deferred to the full version of this paper [1].

3 The Multiparty Protocol

In this section, we describe our construction and prove Theorem 1. This result is formally restated in Section 3.3 (as Corollary 1) and proved therein.

In Section 3.1, we describe a construction of an m-party coin-tossing protocol tolerating up to 2/3 corruptions. In Section 3.2, we describe the main construction of an m-party almost optimally fair coin-tossing protocol tolerating up to 3/4 corruptions.

3.1 A Coin-Tossing Protocol for $t < 2m/3$

The following algorithm, is an extension of the two-party share generator, presented in [24], to the multiparty case.

Algorithm 5 ($\mathbf{MultipartyShareGen}_{<2/3} - \boldsymbol{HG}(\varepsilon, m, t)$)
Let $r \in \mathbb{N}$ be the number of rounds.

Input: *Number of rounds r, $\varepsilon = \varepsilon(n) \in \left[-\frac{1}{2}, \frac{1}{2}\right]$, the number of parties m, and an upper bound t on the number of corrupted parties.*
Denote $h = m - t$. Observe that a subset $J \subset [m]$ of size $2h - 1$, containing all honest parties has an honest majority.

Selecting coins and defenses:

1. *For every $J \subset [m]$ of size $2h - 1$:*
   (a) *Let S^J be a set with $2s_0$ elements from $\{-1, 1\}$, where each element is sampled according to $\mathcal{B}er(\varepsilon)$.*
   (b) *Let A_0^J be a random subset of S^J of size s_0.*
   (c) *Let d_0^J be 1 if $\sum_{a \in A_0^J} a \geq 0$, and 0 otherwise.*
2. *For $i = 1$ to r:*
   (a) *Sample $x_i \leftarrow \mathcal{B}in_{r-i+1,\varepsilon}$.*
   (b) *For every $J \subset [m]$ of size $2h - 1$, we let A_i^J be a random subset of S^J of size s_i.*

(c) *For every* $J \subset [m]$ *of size* $2h-1$, *let* d_i^J *be 1 if* $\sum_{k=1}^{i} x_k + \sum_{a \in A_i^J} a \geq 0$, *and 0 otherwise*.

Sharing the values:

1. *For* $i \in [r]$, *let* $x_i[j]$ *be a share of* x_i *in a* $(t+1)$*-out-of-*m *secret sharing.*
2. *For* $i \in \{0, \ldots, r\}$, $j \in [m]$, *and* $J \subset [m]$ *of size* $2h-1$, *let* $d_i^J[j]$ *be a share of* d_i^J *in a* h*-out-of-*$(2h-1)$ *secret sharing.*
3. *For* $i \in [r]$, $j \in [m]$, $J \subset [m]$ *of size* $2h-1$, *and* $j' \in J$, *let* $d_i^J[j', j]$ *be a share of* $d_i^J[j']$ *in a* $(t+1)$*-out-of-*m *secret sharing, such that party* $P_{j'}$ *is required in order to recover* $d_i^J[j']$. *This can be done with Construction 4.*

Output: *Party* P_j *receives* $d_i^{J'}[j', j]$, $d_0^J[j]$, $x_i[j]$ *for all* $i \in [r]$, $J, J' \subset [m]$ *of size* $2h-1$, $j \in J$, *and* $j' \in J'$.

Protocol 6 (Multiparty$_{<2/3}$ Coin-Toss) *Let* $r \in \mathbb{N}$ *be the number of rounds. Let* $\hat{m}$, *and* $\hat{t}$ *be two constants where* $\hat{m}$ *denotes the number of parties, and* $\hat{t}$ *is an upper bound on the number of corrupted parties.*

Common input: *Number of rounds* r *and output distribution parameter* ε *(jointly reconstructable, possibly unknown to parties).*

Private inputs: *The private inputs of the parties were given to them by an oracle computing* $\mathrm{HG}(\varepsilon, \hat{m}, \hat{t})$ *as defined in Algorithm 5. The input of party* P_j *for* $j \in [\hat{m}]$ *is* $(\boldsymbol{x}_j, \boldsymbol{d}_j)$, *where*

$$\boldsymbol{x}_j = (x_1[j], \ldots, x_r[j]) \text{ and } \boldsymbol{d}_j = (D_0[j], D_1[j], \ldots D_r[j]),$$

where

$$D_i[j] = \left\{ d_i^J[j', j] \mid J \subset [\hat{m}] \wedge |J| = 2h-1 \wedge j' \in J \right\}, \text{ for } i \in [r]$$

and

$$D_0[j] = \left\{ d_0^J[j] \mid J \subset [\hat{m}] \wedge |J| = 2h-1 \wedge j \in J \right\}.$$

Interaction rounds: *For* $i = 1$ *to* r*:*

(a) *Each party* P_j *sends* $d_i^J[j', j]$ *to* $P_{j'}$ *for every* $j' \neq j$ *and* $J \subset [\hat{m}]$ *of size* $2h-1$, *such that* $j' \in J$.

(b) *The parties reconstruct* x_i.

Output: *The honest parties output 1 if* $\sum_{i=1}^{r} x_i \geq 0$, *and outputs 0 otherwise*.

In case of abort: *Let* $J \subset [\hat{m}]$ *be the set of remaining parties. If* $|J| \geq \hat{t}+1$, *then the parties in* J *go on with the execution of the protocol. Otherwise, they reconstruct and output* $d_i^{J'}$, *for the lexicographically first* $J' \subset [\hat{m}]$ *of size* $2h-1$, *such that* $J \subseteq J'$, *and for the largest* i *for which they have all of the corresponding shares (for the parties of* J*).*

3.2 A Coin-Tossing Protocol for $t < 3m/4$

Algorithm 7 (MultipartyShareGen$_{<3/4}$)
Let $r \in \mathbb{N}$ be the number of rounds. Let m be a constant representing the number of parties, and let t be a constant which is a bound on the number of corrupted parties. We denote $h = m - t$ (i.e., a lower bound on the number of honest parties). In the following, we call a subset $J \subset [m]$ protected *if $2h-1 \leq |J| \leq t$.*

Input: *Number of rounds r.*
Selecting coins and defenses:
For $i = 1$ to r:
1. *Sample $x_i \leftarrow \mathcal{B}in_{r-i+1}$.*
2. *Let $\varepsilon_i \in \left[-\frac{1}{2}, \frac{1}{2}\right]$ be such that $\widehat{\mathcal{B}in}_{s_i,\varepsilon}\left(-\sum_{k=1}^{i} x_k\right) = \widehat{\mathcal{B}in}_{s_0,\varepsilon_i}(0)$.*
3. *For every protected $J \subset [m]$, sample $d_i^J \leftarrow \mathrm{HG}(\varepsilon_i, |J|, t - m + |J|)$.*

Sharing the values:
1. *For $i \in [r]$, let $x_i[j]$ be a share of x_i in a $(t+1)$-out-of-m secret sharing.*
2. *For $i \in [r]$, $j \in [m]$, and a protected $J \subset [m]$, let $d_i^J[j]$ be a share of d_i^J in a $(t - m + |J| + 1)$-out-of-$|J|$ secret sharing.*
3. *For $i \in [r]$, $j \in [m]$, a protected $J \subset [m]$, and $j' \in J$, let $d_i^J[j', j]$ be a share of $d_i^J[j']$ in a $(t+1)$-out-of-m secret sharing, such that party $P_{j'}$ is required in order to recover $d_i^J[j']$. This can be done with Construction 4.*

Output: *Party P_j receives $d_i^J[j', j]$, $x_i[j]$ for every $i \in [r]$, $J \subset [m]$, and $j' \in J$.*

We are now ready to describe the actual multiparty coin-tossing protocol. We remark that the protocol is defined in the fail-stop model, where corrupted parties must follow the prescribed protocol, unless they decide to prematurely abort the execution at some point. This is done for the sake of simplicity of presentation and compiling the following protocols so that they tolerate any malicious behavior is done by standard techniques, using signatures.

Protocol 8 (Multiparty$_{<3/4}$ Coin-Toss)

Common input: *Number of rounds r.*
Preprocessing: *Parties run a secure with identifiable abort implementation of Algorithm 7 to obtain their respective outputs. If an abort occurred during the execution, then the remaining parties restart the protocol without the aborting parties.*
Interaction rounds: *For $i = 1$ to r:*
(a) Each party P_j sends $d_i^J[j', j]$ to $P_{j'}$ for every $j' \neq j$ and every protected $J \subset [m]$ such that $j' \in J$.
(b) The parties reconstruct x_i.

Output: *The honest parties output 1 if $\sum_{i=1}^{r} x_i \geq 0$, and outputs 0 otherwise.*

In case of abort: *Let $J \subset [m]$ be the set of remaining active parties. If $|J| \geq t+1$, then the parties in J continue with the execution of the protocol. Assume that $|J| \leq t$. If the abort happened before the execution of Algorithm 7, then the parties run a secure with identifiable abort implementation of Algorithm 5 to obtain their respective outputs, and they execute Protocol 6. If the abort happened during the interaction rounds, then the parties execute Protocol 6 with $d_i^{J'}[j]$ as the private input for P_j, for the lexicographic first $J' \subset [m]$ such that $J \subseteq J'$, and for the largest i for which they have all of the corresponding shares.*[4]

3.3 Stating the Main Results

Theorem 9. *Let m and t be two constants such that $t < 3m/4$. Assuming OT exists, then for every $r \in \mathbb{N}$, Protocol 8 is an r-round m-party $O\left(2^{2^m} \cdot \frac{\log^3 r}{r}\right)$-secure coin-tossing protocol tolerating any fail-stop adversary that corrupts up to t parties, in the $\left(\text{MultipartyShareGen}_{<3/4}, \text{MultipartyShareGen}_{<2/3}\right)$-hybrid model (guaranteeing security with identifiable abort).*

Corollary 1. *Let n be the security parameter, and let m and t be two constants, such that $t < 3m/4$. Assuming OT exists, then for every polynomial $r = r(n)$, there exists an r-round m-party $O\left(2^{2^m} \cdot \frac{\log^3 r}{r}\right)$-secure coin-tossing protocol, against any PPT adversary corrupting up to t parties.*

In order to prove Theorem 9 we first need to show that Protocol 6 is secure. The security of Protocol 6 by itself does not suffice, as in Protocol 8 after an abort, the adversary's view contains some additional information, and so, the following Lemma states that the additional information won't help him to bias the outcome.

Lemma 6. *Let $\varepsilon \in \left[-\frac{1}{2}, \frac{1}{2}\right]$, and let $\hat{m}$ and $\hat{t}$ be two constants, such that $\hat{t} < 2\hat{m}/3$. Then for every $r \in \mathbb{N}$, Protocol 6 is an r-round $\hat{m}$-party $\left(\hat{t}, O\left(2^{2^m} \cdot \frac{\log^3 r}{r}\right)\right)$-unbiased ε-coin-toss protocol tolerating any fail-stop adversary, corrupting up to $\hat{t}$ parties. Moreover, the above holds even when the adversary gets ε as an auxiliary input.*

The proof of Lemma 6 is deferred to the final version of this paper [1]. We now use it in combination with the results of [24] to prove Theorem 9.

Proof (of Theorem 9). Assume without loss of generality that $r \equiv 1 \mod 4$ (otherwise, we set the number of rounds to be the largest $r' < r$ such that

[4] Note that in the case where $|J| \leq 2h - 1$, there is an honest majority, and so, in $\text{MultipartyShareGen}_{<3/4}$ we could have given them a common bit reconstruct with full security. We decided to instruct the parties to execute Protocol 6 for the sake of simplicity.

$r' \equiv 1 \mod 4$). Hence, $s_i(r)$ is odd, and the output of the parties in an honest execution (without aborts) is a uniform bit. We also assume that r is larger than some constant, which will be determined by the analysis, as otherwise the theorem holds trivially.

Let $\mathcal{A}$ be a fail-stop adversary and let $\mathcal{C} \subset [m]$ be the set of parties that $\mathcal{A}$ corrupts. By assumption, it holds that $|\mathcal{C}| < 3m/4$. Let V be the view of the adversary $\mathcal{A}$ in a random execution of Protocol 8. For a round $I \in [r] \times \{(a),(b)\}$ in the outer protocol, let V_I be the view of the adversary in round I and let V_I^- be it's view without the abort (if happened). We show that the protocol is $\left(t, O\left(2^{2^m} \cdot \frac{\log^3 r}{r}\right)\right)$-unbiased, i.e., we show that:

$$\mathop{\mathrm{E}}_{V}\left[\left|\Delta\left(V\right) - \Delta\left(V^-\right)\right|\right] = O\left(2^{2^m} \cdot \frac{\log^3 r}{r}\right). \tag{1}$$

Applying Lemma 1 to Equation (1) yields that the protocol is $\left(|\mathcal{C}|, O\left(2^{2^m} \cdot \frac{\log^3 r}{r}\right) + neg(n)\right)$-secure. We next prove the correctness of Equation (1).

We need to analyze the gain of the adversary by prematurely aborting the execution of the protocol. Recall that to prematurely abort the execution of the outer protocol, the adversary needs to instruct at least $m - t$ parties to abort. Otherwise, the remaining active parties are instructed to go on as usual, and indeed, by the properties of the secret sharing scheme, they are able to go through with reconstructing their appropriate secrets. Namely, upon receiving (in Step a of round i) shares $d_i^J[j, j']$ from at least t parties $P_{j'}$, party P_j is able to reconstruct $d_i^J[j]$ (using its own share of it). Similarly, upon receiving (in Step b of round i) shares $x_i[j']$ from at least t parties $P_{j'}$, party P_j is able to reconstruct x_i.

Assume an abort occurred before the interaction rounds. Moreover, we assume that at most t parties remain active. Then by the description of the protocol, the parties are instructed to run a secure with identifiable abort implementation of Protocol 6, and there is no bias in the samples. Then $\Delta(V) = \Delta(V^-) = \frac{1}{2}$, which yields no advantage to the adversary.

Assume an abort occurred during the interaction rounds. Let $I = (i, (\cdot))$ be the first round for which there is an abort and there are at most t active parties remaining. We define two adversaries $\mathcal{A}_{(a)}$ and $\mathcal{A}_{(b)}$ as follows: $\mathcal{A}_{(a)}$ and $\mathcal{A}_{(b)}$ act exactly as does $\mathcal{A}$, until round I, in which $\mathcal{A}$ decided to abort. If $I = (i, (a))$, then $\mathcal{A}_{(a)}$ aborts at $(i, (a))$, and $\mathcal{A}_{(b)}$ completes the execution honestly without aborting. If $I = (i, (b))$, then $\mathcal{A}_{(a)}$ completes the execution honestly without aborting, and $\mathcal{A}_{(b)}$ aborts at $(i, (b))$. Let $V_I^{(a)}$ and $V_I^{(b)}$ be the view of $\mathcal{A}_{(a)}$ and $\mathcal{A}_{(b)}$, respectively.

Assume that $I = (i, (a))$:

The view of the adversary $\mathcal{A}_{(a)}$ consists of:

$$\{x_1, x_2 \ldots x_{i-1}\} \text{ and } D_i^{\mathcal{C}},$$

where

$$D_i^{\mathcal{C}} = \left\{ d_k^{\mathcal{C}'} : |\mathcal{C}' \cap \mathcal{C}| > t - m + |\mathcal{C}'| \wedge k \leq i \right\},$$

is the set of all the defenses that the adversary can see up to and including round i. In addition, the adversary $\mathcal{A}_{(a)}$ sees many shares that are useless to it. Specifically, the adversary $\mathcal{A}_{(a)}$ holds shares of two different types. The first type are shares of the elements in its view that $\mathcal{A}_{(a)}$ completely reconstructed (i.e., those specified above). This type of shares are useless to $\mathcal{A}_{(a)}$, as they were chosen independently of all other information. The second type are shares of the defense values of other sets that $\mathcal{A}_{(a)}$ cannot reconstruct (since it sees at most t such shares). This type of shares are useless to $\mathcal{A}_{(a)}$ by the properties of secret sharing schemes. We thus, disregard these two types of shares, and continue with the analysis as if the view of $\mathcal{A}_{(a)}$ consists only of the random coins and of $D_i^{\mathcal{C}}$. Formally, the view of the adversary $\mathcal{A}_{(a)}$ may contain only part of $D_i^{\mathcal{C}}$, however, an adversary with more information can always emulate one with less information by simply disregarding parts of its view.
Each $d_k^{\mathcal{C}'}$ is a vector, which consists of $O\left(r^2\right)$ elements from $\{-1, 1\}$, where the elements are sampled according to $\mathcal{B}er(\varepsilon_k)$, where ε_k satisfies $\widehat{\mathcal{B}in}_{s_0,\varepsilon_k}(0) = \widehat{\mathcal{B}in}_{s_k}\left(-\sum_{l=1}^{k} x_l\right)$. As $D_i^{\mathcal{C}}$ has $O\left(r^2\right)$ bits in total, Lemma 2 tell us that:

$$\underset{V_I^{(a)}}{\mathrm{E}}\left[\left|\Delta\left(V_I^{(a)}\right) - \Delta\left(V_I^{(a)-}\right)\right|\right] = O\left(\frac{\log^3 r}{r}\right), \tag{2}$$

Assume that $I = (i, (b))$:
The view of the adversary $\mathcal{A}_{(b)}$ consists of:

$$\{x_1, x_2 \ldots x_i\} \text{ and } D_i^{\mathcal{C}},$$

As in the previous case, we disregard the other shares that $\mathcal{A}_{(b)}$ sees. Since the defenses are sampled independently, given x_i, and since the expectation of each $d_i^{\mathcal{C}'}$ is exactly the game value given $x_1, \ldots x_i$, the adversary gains nothing by aborting in this rounds.

Combining the two cases yields the bound on the maximum bias $\mathcal{A}$ can do in round I:

$$\begin{aligned}
&\underset{V_I}{\mathrm{E}}\left[\left|\Delta\left(V_I\right) - \Delta\left(V_I^{-}\right)\right|\right] \\
&\quad = \underset{V_I^{(a)}}{\mathrm{E}}\left[\left|\Delta\left(V_I^{(a)}\right) - \Delta\left(V_I^{(a)-}\right)\right|\right] + \underset{V_I^{(b)}}{\mathrm{E}}\left[\left|\Delta\left(V_I^{(b)}\right) - \Delta\left(V_I^{(b)-}\right)\right|\right] \\
&\quad = O\left(\frac{\log^3 r}{r}\right).
\end{aligned}$$

In order the conclude the proof of security we need to show that the remaining corrupted parties can't bias the outcome by more than $O\left(\frac{\log^3 r}{r}\right)$. Let $J \subset [m]$

be the set of the remaining parties , let $\hat{m} = |J|$, and let $h = m - t$ be a lower bound on the number of honest parties. Since, at least h parties aborted, it follows that there are at most $\hat{t} := \hat{m} - h$ corrupted parties in J. By assumption, $t < \frac{3m}{4}$, and hence, $\hat{t} < \hat{m} - \frac{m}{4}$. Since $\hat{m} \leq t < \frac{3m}{4}$, it holds that $\hat{t} < \hat{m} - \frac{\hat{m}}{3} = \frac{2\hat{m}}{3}$. Therefore by Lemma 6 it holds that:

$$\mathop{\mathrm{E}}_{V_{\text{inner}}}\left[\left|\Delta\left(V_{\text{inner}}\right) - \Delta\left(V^{-}_{\text{inner}}\right)\right|\right] = O\left(2^{2^m} \cdot \frac{\log^3 r}{r}\right), \tag{3}$$

where V_{inner} is the view of $\mathcal{A}$ in Protocol 6, with ε_i included. Note that Lemma 6 assumes that the adversary's view contains only ε_i as auxiliary input. However, Equation (3) still holds, as the rest of the view is independent of V_I, and give no information to the adversary.

3.3.1 Proof of Corollary 1.

We next sketch the proof of Corollary 1.

Proof Sketch of Corollary 1. We adjust Protocols 6 and 8, so that each message that any of the parties ever needs to send is signed, and all other parties verify this signature upon receiving this messages. If at some point in the execution, party P broadcasts a message that is not properly signed, then, all parties treat this as if P has aborted the computation and is no longer active. This is done similarly to the way presented in [8]. Towards this end, Algorithm 7 is changed so that for every round i, every two parties $P_j, P_{j'}$, and every appropriate subset J, both $x_i[j]$ and $d_i^J[j]$ are signed. In addition, let $\sigma(i, J, j')$ be the signature attributed to $d_i^J[j']$, then, $d_i^J[j', j]$ is redefined to be a share of $\left(d_i^J[j'], \sigma(i, J, j')\right)$ in a $(t+1)$-out-of-m secret sharing, such that party $P_{j'}$ is required in order to recover $d_i^J[j']$. Finally, $d_i^J[j', j]$ is also signed.

We further modify Algorithm 7 so that for every $i \in [r]$, the computation of ε_i (see Item 2) can be done efficiently, similarly to the way done in [24]. Observe that ε_i is only used to sample $O\left(r^2\right)$ independent bits, hence it can be efficiently estimated with $\tilde{\varepsilon}_i$, such that the statistical difference between the samples is bounded by $\frac{1}{r^2}$. It follows that the adjusted Protocol 8 is a r-round, m-party $O\left(2^{2^m} \cdot \frac{\log^3 r}{r} + \frac{r}{r^2}\right)$-secure coin-tossing protocol against any PPT adversary.

Finally, similarly to [8], the modified (efficient) functionality is replaced by a secure with identifiable abort protocol that runs in a constant number of rounds. As explained in [8], thiscan be done using (a variation on) the protocol of [32].

Acknowledgements

We are grateful to Iftach Haitner and Amos Beimel for useful conversations.

Bibliography

[1] B. Alon and E. Omri. Almost-optimally fair multiparty coin-tossing with nearly three-quarters malicious. http://omrier.wixsite.com/eran-omri/almost-opt-fair-multiparty-coin-tos, 2016. Full version of this paper.

[2] G. Asharov. Towards characterizing complete fairness in secure two-party computation. In *Proc. of the Eleventh Theory of Cryptography Conference – TCC 2014*, volume 8349, pages 291–316. Springer, 2014.

[3] G. Asharov, Y. Lindell, and T. Rabin. A full characterization of functions that imply fair coin tossing and ramifications to fairness. In *Proc. of the Tenth Theory of Cryptography Conference – TCC 2013*, volume 7785 of *Lecture Notes in Computer Science*, pages 243–262. Springer, 2013.

[4] G. Asharov, A. Beimel, N. Makriyannis, and E. Omri. Complete characterization of fairness in secure two-party computation of boolean functions. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, pages 199–228, 2015.

[5] Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *Theory of cryptography*, pages 137–156. Springer, 2007.

[6] B. Averbuch, M. Blum, B. Chor, S. Goldwasser, and S. Micali. How to implement Bracha's $O(\log n)$ Byzantine agreement algorithm, 1985. Unpublished manuscript.

[7] A. Beimel, Y. Lindell, E. Omri, and I. Orlov. $1/p$-secure multiparty computation without honest majority and the best of both worlds. In P. Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 277–296. Springer, 2011.

[8] A. Beimel, E. Omri, and I. Orlov. Protocols for multiparty coin toss with dishonest majority. *J. Cryptology*, 28(3):551–600, 2015. Conference version in: T. Rabin, editor, Advances in Cryptology – CRYPTO 2010, volume 6223 of Lecture Notes in Computer Science, pages 538-557. Springer-Verlag, 2010.

[9] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–10, 1988.

[10] Berman, I. Haitner, and A. Tentes. Coin flipping of *any* constant bias implies one-way functions. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 398–407, 2014.

[11] M. Blum. Coin flipping by telephone. In *Advances in Cryptology – CRYPTO '81*, pages 11–15, 1981.

[12] M. Blum. Coin flipping by telephone a protocol for solving impossible problems. *SIGACT News*, 15(1):23–27, 1983.

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

[14] R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 364–369, 1986.

[15] R. Cleve and R. Impagliazzo. Martingales, collective coin flipping and discrete control processes. Manuscript, 1993.

[16] D. Dachman-Soled, Y. Lindell, M. Mahmoody, and T. Malkin. On the black-box complexity of optimally-fair coin tossing. In *Theory of Cryptography, Eighth Theory of Cryptography Conference, TCC 2011*, volume 6597, pages 450–467, 2011.

[17] D. Dachman-Soled, M. Mahmoody, and T. Malkin. Can optimally-fair coin tossing be based on one-way functions? In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 217–239, 2014.

[18] O. Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[19] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *stoc19*, pages 218–229, 1987.

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

[21] S. D. Gordon and J. Katz. Partial fairness in secure two-party computation. *Journal of Cryptology*, 25(1):14–40, 2012.

[22] S. D. Gordon, C. Hazay, J. Katz, and Y. Lindell. Complete fairness in secure two-party computation. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 413–422, 2008.

[23] I. Haitner and E. Omri. Coin Flipping with Constant Bias Implies One-Way Functions. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 110–119, 2011.

[24] I. Haitner and E. Tsfadia. An almost-optimally fair three-party coin-flipping protocol. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 408–416, 2014. URL http://www.cs.tau.ac.il/~iftachh/papers/3PartyCF/QuasiOptimalCF_Full.pdf.

[25] I. Haitner, M. Nguyen, S. J. Ong, O. Reingold, and S. Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM Journal on Computing*, 39(3):1153–1218, 2009.

[26] Y. Ishai, R. Ostrovsky, and V. Zikas. Secure multi-party computation with identifiable abort. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 369–386, 2014.

[27] J. Katz. On achieving the "best of both worlds" in secure multiparty computation. In *STOC07*, pages 11–20, 2007.

[28] H. K. Maji, M. Prabhakaran, and A. Sahai. On the Computational Complexity of Coin Flipping. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 613–622, 2010.

[29] N. Makriyannis. On the classification of finite boolean functions up to fairness. In *Security and Cryptography for Networks – 9th International Conference, SCN 2014*, volume 8642 of *Lecture Notes in Computer Science*, pages 135–154. Springer, 2014.

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

[31] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991. Preliminary version in *CRYPTO'89*.

[32] R. Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 232–241, 2004.

[33] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11): 612–613, 1979.