Fair Coin Flipping: Tighter Analysis and the Many-Party Case

Niv Buchbinder[*] Iftach Haitner[†‡] Nissan Levi[†] Eliad Tsfadia[†]

Abstract

In a multi-party *fair* coin-flipping protocol, the parties output a common (close to) unbiased bit, even when some corrupted parties try to bias the output. In this work we focus on the case of dishonest majority, ie at least half of the parties can be corrupted. [18] [STOC 1986] has shown that in *any* m-round coin-flipping protocol the corrupted parties can bias the honest parties' common output bit by $\Theta(1/m)$. For more than two decades the best known coin-flipping protocols against majority was the protocol of [9] [Manuscript 1985], who presented a t-party, m-round protocol with bias $\Theta(t/\sqrt{m})$. This was changed by the breakthrough result of [40] [TCC 2009], who constructed an m-round, *two*-party coin-flipping protocol with optimal bias $\Theta(1/m)$. Recently, [30] [STOC 14] constructed an m-round, *three*-party coin-flipping protocol with bias $O(\log^3 m/m)$. Still for the case of more than three parties, against arbitrary number of corruptions, the best known protocol remained the $\Theta(t/\sqrt{m})$-bias protocol of [9].

We make a step towards eliminating the above gap, presenting a t-party, m-round coin-flipping protocol, with bias $O(\frac{t^3\cdot 2^t\cdot\sqrt{\log m}}{m^{1/2+1/(2^{t-1}-2)}})$. This improves upon the $\Theta(t/\sqrt{m})$-bias protocol of [9] for any $t \leq 1/2\cdot\log\log m$, and in particular for $t \in O(1)$, this yields an $1/m^{\frac{1}{2}+\Theta(1)}$-bias protocol. For the three-party case, this yields an $O(\sqrt{\log m}/m)$-bias protocol, improving over the the $O(\log^3 m/m)$-bias protocol of [30]. Our protocol generalizes that of [30], by presenting an appropriate "defense protocols" for the remaining parties to interact in, in the case that some parties abort or caught cheating ([30] only presented a two-party defense protocol, which limits their final protocol to handle three parties).

We analyze our new protocols by presenting a new paradigm for analyzing fairness of coin-flipping protocols. We map the set of adversarial strategies that try to bias the honest parties outcome in the protocol to the set of the feasible solutions of a linear program. The gain each strategy achieves is the value of the corresponding solution. We then bound the the optimal value of the linear program by constructing a feasible solution to its dual.

Keywords: coin-flipping; fair computation; stopping time problems

1 Introduction

In a multi-party *fair* coin-flipping protocol, the parties wish to output a common (close to) unbiased bit, even though some of the parties may be corrupted and try to bias the output. More formally, such protocols should satisfy the following two properties: first, when all parties are honest (i.e., follow the prescribed protocol), they all output the *same* bit, and this bit is unbiased (i.e., uniform over $\{0,1\}$). Second, even when some parties are corrupted (i.e., collude and arbitrarily deviate from the protocol), the remaining parties should still output the *same* bit, and this bit should not be too biased (i.e., its distribution should be close to uniform over $\{0,1\}$). We emphasize that unlike weaker variants of coin-flipping protocol known in the literature, the honest parties should **always** output a common bit, regardless of what the corrupted parties do, and in particular they are not allowed to abort if a cheat was detected.

When a majority of the parties are honest, efficient and *completely* fair coin-flipping protocols are known as a special case of secure multi-party computation with an honest majority [13].[1] However, when there is no honest majority, the situation is more complex.

Negative results. Cleve [18] showed that for *any* efficient two-party m-round coin-flipping protocol, there exists an efficient adversarial strategy to bias the output of the honest party by $\Theta(1/m)$. This lower bound extends to the multi-party case, with no honest majority,

[*]Statistics and Operations Research, Tel Aviv university. E-mail:`niv.buchbinder@gmail.com`.

[†]School of Computer Science, Tel Aviv University. E-mail:{`iftachh@cs.tau.ac.il`, `nisnis.levi@gmail.com`, `eliadtsfadia@gmail.com`}. Research supported by ERC starting grant 638121.

[‡]Member of the Israeli Center of Research Excellence in Algorithms (ICORE) and the Check Point Institute for Information Security.

[1]Throughout, we assume a broadcast channel is available to the parties.

via a simple reduction.

Positive results. Awerbuch, Blum, Chor, Goldwasser, and Micali [9] showed, that if one-way functions exist, a simple m-round majority protocol can be used to derive a t-party coin-flipping protocol with bias $\Theta(t/\sqrt{m})$.[2]

For more than two decades, Awerbuch et al.'s protocol was the best known fair coin-flipping protocol (without honest majority), under *any* hardness assumption and for *any* number of parties. In their breakthrough result, Moran, Naor, and Segev [40] constructed an m-round, *two*-party coin-flipping protocol with optimal bias of $\Theta(1/m)$. In a subsequent work, Beimel, Omri, and Orlov [10] extended the result of [40] for the multi-party case in which *less than* $\frac{2}{3}$ of the parties can be corrupted. More specifically, for any $\ell < \frac{2}{3} \cdot t$, they presented an m-round, t-party protocol, with bias $\frac{2^{2^{\ell-t}}}{m}$ against (up to) ℓ corrupted parties. Recently, Haitner and Tsfadia [30] constructed an m-round, *three*-party coin-flipping protocol with bias $O(\log^3 m/m)$ against two corruptions. In a subsequent work, Alon and Omri [3] presented for any $t \in O(1)$, an m-round, t-party coin-flipping protocol, with bias $O(\log^3 m/m)$ against $\ell < 3t/4$ corrupted parties. (All the above results hold under the assumption that oblivious transfer protocols exist.) Yet, for the case of more than three parties (and unbounded number of corruptions), the best known protocol was the $\Theta(t/\sqrt{m})$-bias protocol of [9].

1.1 Our Result

Our main result is a new multi-party coin flipping protocol.

THEOREM 1.1. (MAIN THEOREM, INFORMAL) *Assuming the existence of oblivious transfer protocols, then for any $m = m(n) \leq \operatorname{poly}(n)$ and $t = t(n) \leq o(\log m)$, there exists an m-round, t-party coin-flipping protocol, with bias $O(\frac{t^3 \cdot 2^t \cdot \sqrt{\log m}}{m^{1/2+1/(2^{t-1}-2)}})$ (against $t-1$ corrupted parties).*

The above protocol improves upon the $\Theta(t/\sqrt{m})$-bias protocol of Awerbuch et al. [9] for any $t \leq \frac{1}{2} \cdot \operatorname{loglog} m$. For $t \in O(1)$, this yields an $1/m^{\frac{1}{2}+\Theta(1)}$-bias protocol. For the three-party case, the above yields an $O(\sqrt{\log m}/m)$-bias protocol, improving over the the $O(\log^3 m/m)$-bias protocol of Haitner and Tsfadia [30].

[2]The result of Awerbuch et al. [9] was never published, and it is contributed to them by Cleve [18] who analyzed the two-party case. Cleve [18] analysis extends to the many party case in a straightforward manner. The protocol of [18] is using family of trapdoor permutations, but the latter were only needed to construct commitment schemes, which we currently know how to construct from any one-way function [32, 33, 41].

We analyze the new protocol by presenting a new paradigm for analyzing fairness of coin-flipping protocols. We upper bound the bias of the protocol by upper-bounding the value of a linear program that *characterizes it*: there exists an onto mapping from the set of adversarial strategies that try to bias the honest parties outcome in the protocol, to the set of the program's feasible solutions, such that the gain a strategy achieves is, essentially, the value of the solution of the program it is mapped to. See Section 1.3 for more details.

1.2 The New Multi-Party Fair Coin-Flipping Protocol

Our fair-flipping protocol follows the paradigm of Haitner and Tsfadia [30]. In the following we focus on *fail-stop* adversaries, ones that follow the protocol description correctly and their only adversarial action is abort prematurely (forcing the remaining parties to decide on their common output without them). Compiling a protocol secure against such fail-stop adversaries into a protocol of the same bias that is secure against any polynomial-time adversary, can be done using standard cryptographic tools assuming the existence of *one-way functions*. We also assume that the parties can *securely compute with abort* any efficient functionality — The only information a party obtains from such a computation is its local output (might be a different output per party). The order of which the outputs are given, however, is arbitrary. In particular, a party that aborts after obtaining its own output, might prevent the remaining parties to get their outputs. For every efficient functionality, a constant-round protocol that securely compute it with abort can be constructed using *oblivious transfer* protocol, a commonly used primitives implied by standard cryptographic assumption, e.g., the *decisional quadratic residuosity problem* is hard (over certain groups).

The protocol of Haitner and Tsfadia [30] enhances the basic majority coin-flipping protocol of Awerbuch et al. [9] with *defense protocols* for the remaining parties to interact in, if some of the parties abort. We consider the following generalization of the [30] m-round protocol for arbitrary number of parties. The functionality $\mathsf{Defense}$ and the sub-protocols $\{\Pi^{t'}\}_{t'<t}$ used in the protocol are specified later.

PROTOCOL 1.1. $(\widehat{\Pi}^t = (\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_t))$

For $i = 1$ to m:

1. *Every proper subset of parties $\mathcal{Z} \subsetneq \{\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_t\}$ securely compute $\mathsf{Defense}^{|\mathcal{Z}|}()$. Let $\mathsf{share}^{\#z,\mathcal{Z}}$ be the output party P_z got back from this call.*

2. *The parties securely compute $\mathsf{Coin}()$ that returns a*

common uniform $\{-1,1\}$ *coin* c_i.

Output: All parties output $\mathsf{sign}(\sum_{i=1}^{m} c_i)$.

Abort: Let $\mathcal{Z} \subsetneq \{\mathsf{P}_1,\ldots,\mathsf{P}_t\}$ *be the remaining (non-aborting) parties. To decide on a common output, the parties in* $\mathcal{Z}$ *interact in the "defense" protocol* $\Pi^{|\mathcal{Z}|}$, *where party* P_z *private input is* $\mathsf{share}^{\#z,\mathcal{Z}}$. *Where if* $\mathcal{Z} = \{\mathsf{P}_z\}$ *(i.e., only a single non-aborting party remained), the party* P_z *outputs* $\mathsf{share}^{\#z,\mathcal{Z}}$.

Namely, the parties interact in an m-round majority protocol. Taking an odd value of m yields that if all parties are honest then they all output the same unbiased bit.

To instantiate the above protocol one needs to define the functionality $\mathsf{Defense}^{t'}$ and the protocol $\Pi^{t'}$, for all $t' < t$. But first let's discuss whether we need these functionality and protocols at all. That is, why not simply instruct the remaining parties to re-toss the coin c_i if some parties abort in Step 2 of the i'th round. This is essentially the vanilla protocol of Awerbuch et al. [9], and it is not hard to get convinced that a malicious (fail-stop) party can bias the output of the protocol by $\Theta(1/\sqrt{m})$. To see that, note that the sum of m unbiased $\{-1,1\}$ coins is roughly uniform over $[-\sqrt{m},\sqrt{m}]$. In particular, the probability that the sum is in $\{-1,1\}$ is $\Theta(1/\sqrt{m})$. It follows that if a party aborts after seeing in Step 2 of the first round that $c_1 = 1$, and by that causing the remaining parties to re-toss c_1, it biases the final outcome of the protocol towards 0 by $\Theta(1/\sqrt{m})$.

To improve upon this $1/\sqrt{m}$ barrier, [30] have defined $\mathsf{Defense}$ such that the expected outcome of $\Pi^{t'}(\mathsf{Defense}(1^{t'}))$ equals $\delta_i = \Pr\left[\mathsf{sign}(\sum_j c_j) = 1 | c_1,\ldots,c_i\right]$ for every t'. Namely, the expected outcome of the remaining parties has *not changed*, if some parties abort in Step 2 of the protocol.[3] The above correlation of the defense values returned by $\mathsf{Defense}$ in Step 1 and the value of c_i returned by Coin in Step 2, however, yields that they give some information about c_i, and thus the (only) weak point of the protocol has shifted to Step 1. Specifically, the bias achieved by aborting in Step 1 of round i is the difference between δ_{i-1}, the expected value of the protocol given the coins $c_1,\ldots,c_{i-1}$ flipped in the previous rounds, and the expected outcome of the protocol given these coins and the defense values given to the corrupted parties in Step 1. If done properly, only limited information about c_i is revealed in Step 1, and thus attacking there is not as effective as attacking in Step 2 of the vanilla (no defense) protocol.

For $t = 2$, [30] have set the defense for the remaining party to be a bit b_i that is set to 1 with probability δ_i. Namely, if a party aborts in Step 2 of the i'th round, the other party output 1 with probability δ_i, and 0 otherwise. Since $\mathrm{E}[b_i] = \delta_i$, attacking in Step 2 (in any round) of the protocol is useless. Moreover, since b_i only leaks "limited information" about δ_i (and thus about c_i), it is possible to show that attacking in Step 1 (in any round) biases the protocol by $(\delta_i - \delta_{i-1})^2$ (to compare to the $(\delta_i - \delta_{i-1})$ bias achieved in Step 2 of the vanilla protocol). These observations yield (see Section 1.3) that the protocol's bias is $\mathrm{polylog}(m)/m$.[4] Generalizing the above for even $t = 3$, however, is non-trivial. The defense values of the remaining parties should allow them to interact in a *fair* protocol Π^2 of expected outcome δ_i. Being fair, protocol Π^2 should contain a defense mechanism of its own to avoid one of the remaining parties to bias its outcome by too much (this was not an issue in the the case $t = 1$, in which there is only one remaining party). Actually, designing a $\mathsf{Defense}$ functionality and a protocol Π^2, in a way that the defense values returned by $\mathsf{Defense}$ do not reveal c_i *completely*, and thus turning the protocol to be of bias $1/\sqrt{m}$, is already non-trivial. Yet, [30] managed to find such an implementation of the $\mathsf{Defense}$ functionality and Π^2 that yield a $\mathrm{polylog}(m)/m$-bias protocol.[5] The rather complicated approach use by [30], however, was very tailored for the case that defense sub-protocol Π^2 is a two-party protocol. In particular, it critically relies on the fact that in a two-party protocol there is no defense sub-protocol (rather, the remaining party decides on its output by its own). We take a very different approach to implement the $\mathsf{Defense}$ functionality and its accompanied defense protocol $\Pi^{t'}$.[6]

Algorithm 1.1. (The $\mathsf{Defense}$ functionality)

Input: $1^{t'}$

//Recall that $c_1,\ldots,c_{i-1}$ are the coins flipped in the previous rounds, c_i is the coin to be output in this round call to Coin, and $\delta_i = \Pr\left[\mathsf{sign}(\sum_j c_j) = 1 | c_1,\ldots,c_i\right]$.

[3] The above definition requires $\mathsf{Defense}$ to share a (secret) state with Coin, since both functionalities are defined with respect to the same coin c_i. This non-standard requirement is only for the sake of presentation, and in the actual protocol we replace it with stateless functionalities that share, and maintain, their "state" by secret sharing it between the parties. See Section 3.

[4] More precisely, this bound was proved for the weighted variant of the above protocol, where in round i the functionality Coin returns the sum of $m-i+1$ independent coins. See Section 3.

[5] The analysis we employ in this paper, see Section 1.3, shows that the bias of (a simple variant of) the [30] protocols is actually $\sqrt{\log m}/m$.

[6] For subsets of size two, we are still using the mechanism of [30] that handles such subsets better. We ignore this subtlety for the sake of the introduction.

1. Let $\delta'_i = \delta_i + Noise$, where $Noise$ is random variable of expectation 0.

2. Let $\mathsf{share}^{\#1}, \ldots, \mathsf{share}^{\#t'}$ be t'-out-of-t' secret sharing of δ'_i.[7] Return $\mathsf{share}^{\#i}$ to the i'th party.

Namely, Defense computes a noisy version of δ_i and secret-share the result between the calling parties.

Protocol 1.2. ($\Pi^{t'} = (\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_{t'})$)

Input: Party P_z*'s input is* $\mathsf{share}^{\#z}$.

1. *Each party* P_z *sends its input* $\mathsf{share}^{\#z}$ *to the other parties, and all parties set* $\delta' = \bigoplus_{z=1}^{t'} \mathsf{share}^{\#z}$.

2. *Continue as the* δ'*-biased version of Protocol* $\widehat{\Pi}^{t'}$:

   - Coin *sets the coin* c_i *to be* 1 *with probability* $1/2+\varepsilon$ *(rather than* $1/2$*), for* $\varepsilon \in [-1,1]$ *being the value such that* δ' *is the probability that the sum of* m *independent* $(1/2+\varepsilon)$*-biased* $\{-1,1\}$ *coins is positive.*

   - *The definition of* Defense *is changed accordingly to reflect this change in the bias of the coins.*

Since $\mathrm{E}[Noise] = 0$, the expected outcome of $\Pi^{t'}(\mathsf{Defense}(1^{t'}))$ is indeed δ_i. Note that since the corrupted parties can use their shares to reconstruct the value of δ'_i sampled in the all-corrupted calls to Defense (those calls made by subsets in which all parties are corrupted), the values returned by Defense do leak some information about δ_i, and thus about the coin c_i. But if $Noise$ is "noisy enough" (i.e., high enough variance), then δ'_i does not leak too much information about δ_i. Hence, by taking noisy enough $Noise$, we make $\widehat{\Pi}^t$ robust against a single abort (this is similar to the two-party protocol). On its second abort, however, an attacker is actually attacking the above sub-protocol $\Pi^{t'}$, which provides the attacker a very effective attack opportunity: the attacker who is first to reconstruct $\delta' = \delta'_i$, can choose to abort and by that make the remaining parties to continue with an execution whose expected outcome is δ_i. Hence, it can bias the protocol's outcome by $\delta_i - \delta'_i$. If δ'_i is with high probability far from δ_i, this makes the resulting protocol unfair. A partial solution for this problem is to defend the reconstruction of δ' in a similar way to how we defend the reconstruction of the coin c_i; before reconstruction the value of δ', call (a variant of) Defense to defend the parties in the case an abort happens in the reconstruction step. As in the two-party protocol we mentioned before, the use of Defense reduces the bias from $\delta_i - \delta'_i$ to $(\delta_i - \delta'_i)^2$. This limitation dictates us to choose $Noise$ of bounded variance. But when using such a $Noise$ function, we are no longer in the situation where δ'_i does not leak significant information about δ_i, making the protocol $\widehat{\Pi}^t$ vulnerable to aborting attacks. Actually, the situation is even more complicated, non-optimal t-party protocol $\widehat{\Pi}^t$ makes the sub-protocols used when an abort occurs even less optimal. The solution is to choose a variance of $Noise$ that compromises between these two contradicting requirements. For not too large t, the right choice of parameters yields a protocol of the claimed biased, significantly improving over the $(1/\sqrt{m})$-bias vanilla protocol.

1.3 Proving Fairness via Linear Program

The security of (any instantiation) of Protocol 1.1 described in the previous section, easily reduced to the value of the appropriate *online binomial game*, first considered by [30].

Online binomial games. An m-round online-binomial game is a game between the (honest, randomized) challenger and an all-powerful player. The game is played for m rounds. At round i, the challenger tosses an independent $\{-1,1\}$ coin c_i. The final outcome of the game is set to one if the overall sum of the coins is positive. Following each round, the challenger sends the player some information (i.e., *hint*) about the outcome of the coins tossed until this round. After getting the hint, the player decides whether to *abort*, or to continue playing. If aborts, the game stops and the player is rewarded with δ_{i-1} — the probability that the output of the game is one given coins $c_1, \ldots, c_{i-1}$. If never aborted, the player is rewarded with the (final) outcome of the game.[8] The bias of an m-round game G_m, denoted $\mathsf{Bias}(\mathsf{G}_m)$, is the advantage the *best* all-powerful player achieves over the passive (non-aborting) player, namely its expected reward minus $\frac{1}{2}$.

The connection between such line Binomial games and the coin-flipping protocols $\widehat{\Pi}^t$ described in the previous section is rather straightforward. Recall that an adversary controlling some of the parties in an execution of Protocol $\widehat{\Pi}^t$ gains nothing by aborting in Step 2, and thus we can assume without loss of generality that it only aborts, if ever, at Step 1 of some

[7] I.e., $\{\mathsf{share}^{\#i}\}$ are uniform strings conditioned on $\bigoplus_{i=1}^{t'} \mathsf{share}^{\#i} = \delta'_i$. (We assume for simplicity that δ'_i has a short binary representation.)

[8] A different and somewhat more intuitive way to think about the game is as follows. In each round, after getting the hint, the player can instruct the challenger to *re-toss* the current round coin, but it can only do that at most once during the duration of the game. After the game ends, the player is rewarded with its final outcome.

rounds.[9] Recall that the gain achieved from aborting in Step 1 of round i is the difference between δ_{i-1}, here the expected outcome of the protocol given the coins $c_1, \ldots, c_{i-1}$ flipped in the previous rounds, and the expected outcome of the protocol given these coins, and the defense values given to the corrupted parties in Step 1. It follows that the maximal bias obtained by a *single* abort, is exactly the bias of the online binomial game, in which the hints are set to the defense values of the corrupted parties. The biased achieved by t aborts in the protocol, is at most t times the bias of the corresponding game.

Bounding online binomial games via a linear program. Upperbounding the bias of even a rather simple binomial game is not easy. In general, our problem fits in the well studied area of *stopping-time problems*, and our goal is to upperbound the value of the optimal stopping strategy. Specifically, in our problem it is hard to take advantage of the fact that the player does not know beforehand which round will yield him the largest gain. A pessimistic approach, taken in [30], is to consider non-adaptive players that can only abort in a predetermined round, and then upperbound general players using a union bound. This approach effectively assumes the player is told the round it is best to abort, and as we prove here misses the right bound by a polylog factor. We take a different approach. The idea is to map the set of all possible strategies of the game into feasible solutions to a linear formulation. The bias each strategy achieves is equal to the objective value of the corresponding solution. We then use duality to bound the maximal value of the LP. Interestingly, we also prove the other direction: each feasible solution to the LP corresponds to a possible strategy of the game. This shows that bounding the value of the linear formulation is actually equivalent to bounding the value of the best strategy in the game.

The intuition of the linear formulation is simple. For a given binomial game we consider all possible states. Specifically, each state u is characterized by the the current round, i, the sum of coins tossed so far (in the first $i-1$ rounds), b, and the hint h given to the strategy. We use the notation $u = \langle i, b, h \rangle$. For each state u, let p_u be the probability that the game visits state u. For two nodes u, v let $p_{v|u}$ be the probability that the game visits v given that it visits state u. We also write $u < v$ to indicate that the round of v is strictly larger than the round of u. For a given state v let c_v be the expected outcome of the game given that the strategy aborts at state v. Given a strategy S, let a_v^{S} be the marginal probability that the strategy aborts at state v. It is easy to see that the bias achieved by strategy S can be written as:

$$\mathsf{Bias}_m(\mathsf{S}) = \sum_{v \in \mathcal{U}} a_v^{\mathsf{S}} \cdot c_v - \frac{1}{2}$$

Next, we build a linear formulation whose variables are the marginal probabilities a_v, capturing the probability that a strategy aborts at state v. One clear constraint on the variables is that the variables a_v are non-negative. Another obvious constraint is that a_v can never be larger than p_v, the probability that the game visits state v. A more refined constraint is $a_v + \sum_{u|u<v} a_u \cdot p_{v|u} \leq p_v$. Intuitively, this constraint stipulates that the marginal probability of aborting at state v plus the probability that the game visits v and the strategy aborted in a state $u < v$, cannot exceed the probability that the game visits v. We prove that this is indeed a valid constraint for any strategy and also that any solution that satisfies this constraint can be mapped to a valid strategy. As all our constraints and the objective function are linear, this gives us a linear program that characterizes all strategies.

Formulating the linear program is just the first step. Although there are many methods for solving (exactly) a specific linear program, we are interested in (bounding) the *asymptotic* behavior of the optimal solution of the program as a function of m. To bound the solution we construct an asymptotic feasible solution to the dual program. This gives (by weak duality) an upper bound on the optimal bias obtained by any strategy.

1.4 Additional Related Work

Cleve and Impagliazzo [19] showed that in the *fail-stop model* any two-party m-round coin-flipping protocol has bias $\Omega(1/\sqrt{m})$; adversaries in this model are computationally unbounded, but they must follow the instructions of the protocol, except for being allowed to abort prematurely. Dachman-Soled et al. [20] showed that the same holds for $o(n/\log n)$-round protocols in the random-oracle model — the parties have oracle access to a uniformly chosen function over n bit strings.

There is a vast literature concerning coin-flipping protocols with weaker security guarantees. Most notable among these are protocols that are *secure with abort*. According to this security definition, if a cheat is detected or if one of the parties aborts, the remaining parties are not required to output anything. This form of security is meaningful in many settings, and it is typically much easier to achieve; assuming one-way functions exist, secure-with-abort protocols of negligi-

[9]Actually, in an inner sub-protocols $\Pi^{t'}$ the attacker can also aborts in the steps where δ' is reconstructed. But bounding the effect of such aborts is rather simple, comparing to those done is Step 1, and we ignore such aborts from the current discussion.

ble bias are known to exist against any number of corrupted parties [15, 31, 42]. To a large extent, one-way functions are also necessary for such coin-flipping protocols [14, 28, 35, 38].

Coin-flipping protocols were also studied in a variety of other models. Among these are collective coin-flipping in the *perfect information model*: parties are computationally unbounded and all communication is public [4, 12, 22, 44, 45], and protocols based on physical assumptions, such as quantum computation [1, 5, 6] and tamper-evident seals [39].

Perfectly fair coin-flipping protocols (i.e., zero bias) are a special case of protocols for *fair* secure function evaluation (SFE). Intuitively, the security of such protocols guarantees that when the protocol terminates, either everyone receives the (correct) output of the functionality, or no one does. While Cleve [18]'s result yields that some functions do not have fair SFE, it was recently shown that many interesting function families do have (perfectly) fair SFE [26, 7, 8].

1.5 Open Problems

Finding the the optimal bias t-party coin-flipping protocol for $t > 2$, remained the main open question in this area. While the gap between the upper and lower bound for the three-party case is now quite small (i.e., $O(\sqrt{\log m})$ factor), the gap for $t > 3$ is still rather large, and for $t > \log\log m/2$ the best protocol remained the $t/\sqrt{m}$-bias protocol of [9].

Acknowledgment We thank Eran Omri for very useful discussions.

Paper Organization Notations and the definitions used throughout the paper are given in Section 2. Our coin-flipping protocol along with its security proof are given in Section 3. The proofs given in Section 3 rely of claims and lemmas omitted in this version. For more details refer to Buchbinder et al. [16].

2 Preliminaries

2.1 Notation

We use calligraphic letters to denote sets, uppercase for random variables and functions, lowercase for values, boldface for vectors and capital boldface for matrices. All logarithms considered here are in base two. For a vector v, we denote its i-th entry by v_i or $v[i]$. For $a \in \mathbb{R}$ and $b \geq 0$, let $a \pm b$ stand for the interval $[a-b, a+b]$. Given sets $\mathcal{S}_1, \ldots, \mathcal{S}_k$ and k-input function f, let $f(\mathcal{S}_1, \ldots, \mathcal{S}_k) := \{f(x_1, \ldots, x_j) \colon x_i \in \mathcal{S}_i\}$, e.g., $f(1 \pm 0.1) = \{f(x) \colon x \in [.9, 1.1]\}$. For $n \in \mathbb{N}$, let $[n] := \{1, \ldots, n\}$ and $(n) := \{0, \ldots, n\}$. Given a vector $v \in \{-1,1\}^*$, let $w(v) := \sum_{i \in [|v|]} v_i$. Given a vector $v \in \{-1,1\}^*$ and a set of indexes $\mathcal{I} \subseteq [|v|]$, let $v_{\mathcal{I}} = (v_{i_1}, \ldots, v_{i_{|\mathcal{I}|}})$ where $i_1, \ldots, i_{|\mathcal{I}|}$ are the ordered elements of $\mathcal{I}$. We let the XOR of two integers, stands for the *bitwise* XOR of their bit representations, and we let $\mathsf{sign} \colon \mathbb{R} \mapsto \{0,1\}$ be the function that outputs one on non-negative input and zero otherwise.

Distributions. Given a distribution D, we write $x \leftarrow D$ to indicate that x is selected according to D. Similarly, given a random variable X, we write $x \leftarrow X$ to indicate that x is selected according to X. Given a finite set $\mathcal{S}$, we let $s \leftarrow \mathcal{S}$ denote that s is selected according to the uniform distribution on $\mathcal{S}$. The support of a distribution D over a finite set $\mathcal{U}$, denoted $\mathrm{Supp}(D)$, is defined as $\{u \in \mathcal{U} : D(u) > 0\}$. The *statistical distance* of two distributions P and Q over a finite set $\mathcal{U}$, denoted as $\mathrm{SD}(P,Q)$, is defined as $\max_{\mathcal{S} \subseteq \mathcal{U}} |P(\mathcal{S}) - Q(\mathcal{S})| = \frac{1}{2} \sum_{u \in \mathcal{U}} |P(u) - Q(u)|$.

For $\delta \in [0,1]$, let $\mathcal{B}er(\delta)$ be the Bernoulli probability distribution over $\{0,1\}$, taking the value 1 with probability δ and 0 otherwise. For $\varepsilon \in [-1,1]$, let $\mathcal{C}_\varepsilon$ be the Bernoulli probability distribution over $\{-1,1\}$, taking the value 1 with probability $\frac{1}{2}(1+\varepsilon)$ and -1 otherwise. For $n \in \mathbb{N}$ and $\varepsilon \in [-1,1]$, let $\mathcal{C}_{n,\varepsilon}$ be the binomial distribution induced by the sum of n independent random variables, each distributed according to $\mathcal{C}_\varepsilon$. For $n \in \mathbb{N}$, $\varepsilon \in [-1,1]$ and $k \in \mathbb{Z}$, let $\widehat{\mathcal{C}}_{n,\varepsilon}(k) := \Pr_{x \leftarrow \mathcal{C}_{n,\varepsilon}}[x \geq k] = \sum_{t=k}^{n} \mathcal{C}_{n,\varepsilon}(t)$. For $n \in \mathbb{N}$ and $\delta \in [0,1]$, let $\widehat{\mathcal{C}}_n^{-1}(\delta)$ be the value $\varepsilon \in [-1,1]$ with $\widehat{\mathcal{C}}_{n,\varepsilon}(0) = \delta$. For $n \in \mathbb{N}$, $\ell \in [n]$ and $p \in \{-n, \ldots, n\}$, define the hyper-geometric probability distribution $\mathcal{HG}_{n,p,\ell}$ by $\mathcal{HG}_{n,p,\ell}(k) := \Pr_{\mathcal{I}}[w(v_{\mathcal{I}}) = k]$, where $\mathcal{I}$ is an ℓ-size set uniformly chosen from $[n]$ and $v \in \{-1,1\}^n$ with $w(v) = p$. Let $\widehat{\mathcal{HG}}_{n,p,\ell}(k) := \Pr_{x \leftarrow \mathcal{HG}_{n,p,\ell}}[x \geq k] = \sum_{t=k}^{\ell} \mathcal{HG}_{n,p,\ell}(t)$. Let $\Phi \colon \mathbb{R} \mapsto (0,1)$ be the cumulative distribution function of the standard normal distribution, defined by $\Phi(x) := \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-\frac{t^2}{2}} dt$. Finally, for $n \in \mathbb{N}$ and $i \in [n]$, let $\ell_n(i) := (n+1-i)^2$ and $\mathsf{sum}_n(i) := \sum_{j=i}^{n} \ell_n(j)$. We summarize the different notations used throughout the paper in the following tables.

2.2 Facts About the Binomial Distribution

Fact 2.1. (Hoeffding's inequality for $\{-1,1\}$) *Let $n, t \in \mathbb{N}$ and $\varepsilon \in [-1,1]$. Then*

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

Proof. Immediately follows by [34].

The following proposition is proven in [29].

Proposition 2.1. *Let $n \in \mathbb{N}$, $t \in \mathbb{Z}$ and $\varepsilon \in [-1,1]$ be such that $t \in \mathrm{Supp}(\mathcal{C}_{n,\varepsilon})$, $|t| \leq n^{\frac{3}{5}}$ and $|\varepsilon| \leq n^{-\frac{2}{5}}$. Then*

$$\mathcal{C}_{n,\varepsilon}(t) \in (1 \pm \mathsf{error}) \cdot \sqrt{\frac{2}{\pi}} \cdot \frac{1}{\sqrt{n}} \cdot e^{-\frac{(t-\varepsilon n)^2}{2n}},$$

for $\mathsf{error} = \xi \cdot (\varepsilon^2 |t| + \frac{1}{n} + \frac{|t|^3}{n^2} + \varepsilon^4 n)$ *and a universal constant* ξ.

For the proof of the following propositions refer to Buchbinder et al. [16].

Proposition 2.2. *Let* $n \in \mathbb{N}$, $\varepsilon \in [-1,1]$ *and let* $\mu := \mathrm{E}_{x \leftarrow \mathcal{C}_{n,\varepsilon}}[x] = \varepsilon \cdot n$. *Then for every* $k > 0$ *it holds that*

1. $\mathrm{E}_{x \leftarrow \mathcal{C}_{n,\varepsilon} | |x-\mu| \le k}\left[(x-\mu)^2\right] \le \mathrm{E}_{x \leftarrow \mathcal{C}_{n,\varepsilon}}\left[(x-\mu)^2\right] \le n$.
2. $\mathrm{E}_{x \leftarrow \mathcal{C}_{n,\varepsilon} | |x-\mu| \le k}\left[|x-\mu|\right] \le \mathrm{E}_{x \leftarrow \mathcal{C}_{n,\varepsilon}}\left[|x-\mu|\right] \le \sqrt{n}$.

Proposition 2.3. *Let* $n, n' \in \mathbb{N}$, $k \in \mathbb{Z}$, $\varepsilon \in [-1,1]$ *and* $\lambda > 0$ *be such that* $n \le n'$, $|k| \le \lambda \cdot \sqrt{n \log n}$, $|\varepsilon| \le \lambda \cdot \sqrt{\frac{\log n}{n}}$, *and let* $\delta = \widehat{\mathcal{C}}_{n,\varepsilon}(k)$. *Then*

$$\widehat{\mathcal{C}}_{n'}^{-1}(\delta) \in \frac{\varepsilon n - k}{\sqrt{n \cdot n'}} \pm \mathsf{error},$$

for $\mathsf{error} = \varphi(\lambda) \cdot \frac{\log^{1.5} n}{\sqrt{n \cdot n'}}$ *and a universal function* φ.

2.3 Facts About the Hypergeometric Distribution

Fact 2.2. (Hoeffding's inequality) *Let* $\ell \le n \in \mathbb{N}$, *and* $p \in \mathbb{Z}$ *with* $|p| \le n$. *Then*

$$\Pr_{x \leftarrow \mathcal{HG}_{n,p,\ell}}\left[|x-\mu| \ge t\right] \le e^{-\frac{t^2}{2\ell}},$$

for $\mu = \mathrm{E}_{x \leftarrow \mathcal{HG}_{n,p,\ell}}[x] = \frac{\ell \cdot p}{n}$.

Proof. Immediately follows by [46, Equations (10),(14)].

For the proof of the following propositions refer to Buchbinder et al. [16].

Proposition 2.4. *Let* $n \in \mathbb{N}$, $\ell \in [\lfloor \frac{n}{2} \rfloor]$, $p, t \in \mathbb{Z}$ *and* $\lambda > 0$ *be such that* $|p| \le \lambda \cdot \sqrt{n \log n}$, $|t| \le \lambda \cdot \sqrt{\ell \log \ell}$ *and* $t \in \mathrm{Supp}(\mathcal{HG}_{n,p,\ell})$. *Then*

$$\mathcal{HG}_{n,p,\ell}(t) = (1 \pm \mathsf{error}) \cdot \sqrt{\frac{2}{\pi}} \cdot \frac{1}{\sqrt{\ell(1-\frac{\ell}{n})}} \cdot e^{-\frac{(t-\frac{p\ell}{n})^2}{2\ell(1-\frac{\ell}{n})}},$$

for $\mathsf{error} = \varphi(\lambda) \cdot \frac{\log^{1.5} \ell}{\sqrt{\ell}}$ *and a universal function* φ.

Proposition 2.5. *Let* $n \in \mathbb{N}$, $\ell \in [\lfloor \frac{n}{2} \rfloor]$, $p, k \in [n]$ *and* $\lambda > 0$ *be such that* $|p| \le \lambda \cdot \sqrt{n \log n}$ *and* $|k| \le \lambda \cdot \sqrt{\ell \log \ell}$. *Then*

$$\widehat{\mathcal{HG}}_{n,p,\ell}(k) \in \Phi\left(\frac{k - \frac{p \cdot \ell}{n}}{\sqrt{\ell(1-\frac{\ell}{n})}}\right) \pm \mathsf{error},$$

where $\mathsf{error} = \varphi(\lambda) \cdot \frac{\log^{1.5} \ell}{\sqrt{\ell}}$ *for some universal function* φ.

Proposition 2.6. *Let* $n \in \mathbb{N}$, $\ell \in [\lfloor \frac{n}{2} \rfloor]$, $p, k \in [n]$ *and* $\lambda > 0$ *be such that* $|p| \le \lambda \cdot \sqrt{n \log n}$ *and* $|k| \le \lambda \cdot \sqrt{\ell \log \ell}$ *and let* $\delta = \widehat{\mathcal{HG}}_{n,p,\ell}(k)$. *Then for every* $m \ge \ell$ *it holds that*

$$\widehat{\mathcal{C}}_m^{-1}(\delta) \in \frac{\frac{p \cdot \ell}{n} - k}{\sqrt{m \cdot \ell(1-\frac{\ell}{n})}} \pm \mathsf{error},$$

where $\mathsf{error} = \varphi(\lambda) \cdot \frac{\log^{1.5} \ell}{\sqrt{m \cdot \ell}}$ *for some universal function* φ.

2.4 Multi-Party Computation

2.4.1 Protocols

The following discussion is restricted to no private input protocols (such restricted protocols suffice for our needs). A t-party protocol is defined using t Turing Machines (TMs) $\mathsf{P}_1, \ldots, \mathsf{P}_t$, having the security parameter 1^κ as their common input. In each round, the parties broadcast and receive messages on a broadcast channel. At the end of protocol, each party outputs some binary string. The parties communicate in a synchronous network, using only a broadcast channel: when a party broadcasts a message, all other parties see *the same* message. This ensures some consistency between the information the parties have. There are no private channels and all the parties see all the messages, and can identify their sender. We do not assume simultaneous broadcast. It follows that in each round, some parties might hear the messages sent by the other parties before broadcasting their messages. We assume that if a party aborts, it first broadcasts the message Abort to the other parties, and without loss of generality only does so at the end of a round in which it is supposed to send a message. A protocol is *efficient*, if its parties are PPTM, and the protocol's number of rounds is a computable function of the security parameter.

This work focuses on efficient protocols, and on malicious, non-adaptive PPT adversaries for such protocols. An adversary is allowed to corrupt some subset of the parties; before the beginning of the protocol, the adversary corrupts a subset of the parties that from now on may arbitrarily deviate from the protocol. Thereafter, the adversary sees the messages sent to the corrupted parties and controls their messages. We also consider the so called *fail-stop* adversaries. Such adversaries follow the prescribed protocol, but might abort prematurely. Finally, the honest parties follow the instructions of the protocol to its completion.

2.4.2 The Real vs. Ideal Paradigm

The security of multi-party computation protocols is defined using the *real* vs. *ideal* paradigm [17, 24]. In this paradigm, the *real-world model*, in which protocols is executed is compared to an *ideal model* for executing the task at hand. The latter model involves a trusted party whose functionality captures the security requirements of the task. The security of the real-world protocol is argued by showing that it "emulates" the ideal-world protocol, in the following sense: for any real-life adversary A, there exists an ideal-model adversary (also known as simulator) $\mathbb{A}$ such that the global output of an execution of the protocol with A in the real-world model is distributed similarly to the global output of running $\mathbb{A}$ in the ideal model. The following discussion is restricted to random, no-input functionalities. In addition, to keep the presentation simple, we limit our attention to uniform adversaries.[10]

The Real Model. Let π be an t-party protocol and let A be an adversary controlling a subset $\mathcal{C} \subseteq [t]$ of the parties. Let $\mathrm{REAL}_{\pi,\mathsf{A},\mathcal{C}}(\kappa)$ denote the output of A (i.e., without loss of generality its view: its random input and the messages it received) and the outputs of the honest parties, in a random execution of π on common input 1^κ. Recall that an adversary is *fail stop*, if until they abort, the parties in its control follow the prescribed protocol (in particular, they property toss their private random coins). We call an execution of π with such a fail-stop adversary, a fail-stop execution.

The Ideal Model. Let f be a t-output functionality. If f gets a security parameter (given in unary), as its first input, let $f_\kappa(\cdot) = f(1^\kappa, \cdot)$. Otherwise, let $f_\kappa = f$. An ideal execution of f with respect to an adversary $\mathbb{A}$ controlling a subset $\mathcal{C} \subseteq [t]$ of the "parties" and a security parameter 1^κ, denoted $\mathrm{IDEAL}_{f,\mathbb{A},\mathcal{C}}(\kappa)$, is the output of the adversary $\mathbb{A}$ and that of the trusted party, in the following experiment.

Experiment 2.1.

1. *The trusted party sets* $(y_1, \ldots, y_t) = f_\kappa(X)$, *where* X *is a uniform element in the domain of* f_κ, *and sends* $\{y_i\}_{i \in \mathcal{C}}$ *to* $\mathbb{A}(1^\kappa)$.

2. $\mathbb{A}(1^\kappa)$ *sends the message* Continue/ Abort *to the trusted party, and locally outputs some value.*

3. *The trusted party outputs* $\{o_i\}_{i \in [t] \setminus \mathcal{C}}$, *for* o_i *being* y_i *if* $\mathbb{A}$ *instructs* Continue, *and* $\perp$ *otherwise.*

. . . .

An adversary $\mathbb{A}$ is non-aborting, if it never sends the Abort message.

α-secure computation. The following definitions adopts the notion of α-secure computation [11, 25, 37] for our restricted settings.

Definition 2.2. (α-secure computation) *An efficient t-party protocol π computes a t-output functionality f in a α-*secure *manner [resp., against fail-stop adversaries], if for every $\mathcal{C} \subsetneq [t]$ and every [resp., fail-stop]* PPT *adversary A controlling the parties indexed by $\mathcal{C}$,*[11] *there exists a* PPT *$\mathbb{A}$ controlling the same parties, such that*

$$\mathrm{SD}\left(\mathrm{REAL}_{\pi,\mathsf{A},\mathcal{C}}(\kappa), \mathrm{IDEAL}_{f,\mathbb{A},\mathcal{C}}(\kappa)\right) \leq \alpha(\kappa),$$

for large enough κ. A protocol securely compute a functionality f, if it computes f in a $\mathrm{neg}(\kappa)$-secure manner. The protocol π computes f in a simultaneous α-secure *manner, if the above is achieved by a* non-aborting *$\mathbb{A}$.*

Note that being simultaneous α-secure is a very strong requirement, as it dictates that the cheating real adversary has no way to prevent the honest parties from getting their part of the output, and this should be achieved with no simultaneous broadcast mechanism.

2.4.3 Fair Coin-Flipping Protocols

Definition 2.3. (α-fair coin-flipping) *For $t \in \mathbb{N}$ let* $\mathsf{CoinFlip}_t$ *be the t-output functionality from $\{0,1\}$ to $\{0,1\}^t$, defined by* $\mathsf{CoinFlip}_t(b) = b \ldots b$ *(t times). A t-party protocol π is* α-fair coin-flipping protocol, *if it computes* $\mathsf{CoinFlip}_t$ *in a* simultaneous α-secure *manner.*

Proving fairness. Haitner and Tsfadia [29] gave an alternative characterization of fair coin-flipping protocols against fail-stop adversaries. Specifically, Lemma 2.1 below reduces the task of proving fairness of a coin-flipping protocol, against fail-stop adversaries, to proving the protocol is correct: the honest parties always output the same bit, and this bit is uniform in an all honest execution, and to proving the protocol is unbiased: a fail-stop adversary cannot bias the output of the honest parties by too much.

Definition 2.4. (correct coin-flipping protocols) *A protocol is a* correct coin flipping, *if*

- *When interacting with an fails-stop adversary controlling a subset of the parties, the honest parties* always *output the same bit, and*
- *The common output in a random* honest *execution of π, is uniform over $\{0,1\}$.*

[10] All results stated in this paper, straightforwardly extend to the non-uniform settings.

[11] The requirement that $\mathcal{C}$ is a *strict* subset of $[t]$, is merely for notational convinced.

Given a partial view of a fail-stop adversary, we are interesting in the expected outcome of the parties, conditioned on this and the adversary making no further aborts.

Definition 2.5. (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 $\mathsf{C}_\pi(v)$ 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 $\mathsf{C}_\pi(v)$). Let $\mathsf{val}_\pi(v) = \mathrm{E}_{v' \leftarrow \mathsf{C}_\pi(v)}[\mathrm{out}(v')]$, where $\mathrm{out}(v')$ is the common output of the non-aborting parties in v'.*

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

Definition 2.6. (α-unbiased CF protocols) *A t-party, m-round protocol π is* α-unbiased, *if the following holds for every fail-stop adversary A controlling the parties indexed by a subset $\mathcal{C} \subset [t]$ (the corrupted parties). Let V be the corrupted parties' view in a random execution of π in which A controls those parties, and let I_j be the index of the j'th round in which A sent an abort message (set to $m+1$, if no such round). Let V_i be the prefix of V at the end of the i'th round, letting V_0 being the empty view, and let V_i^- be the prefix of V_i with the i'th round abort messages (if any) removed. Then*

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

where $\mathsf{val} = \mathsf{val}_\pi$ is according to Definition 2.5.

Lemma 2.1. ([29], Lemma 2.18) *Let π be a correct, α-unbiased coin-flipping protocol with $\alpha(\kappa) \leq \frac{1}{2} - \frac{1}{p(\kappa)}$, for some $p \in \mathrm{poly}$, then π is a $(\alpha(\kappa) + \mathrm{neg}(\kappa))$-secure coin-flipping protocol against fail-stop adversaries.*

2.4.4 Oblivious Transfer

Definition 2.7. *The $\binom{1}{2}$ oblivious transfer (OT for short) functionality, is the two-output functionality f over $\{0,1\}^3$, defined by $f(\sigma_0, \sigma_1, i) = ((\sigma_0, \sigma_1), (\sigma_i, i))$.*

Protocols the securely compute OT, are known under several hardness assumptions (cf., [2, 21, 23, 27, 36, 43]).

2.4.5 f-Hybrid Model Let f be a t-output functionality. The f-hybrid model is identical to the real model of computation discussed above, but in addition, each t-size subset of the parties involved, has access to a trusted party realizing f. It is important to emphasize that the trusted party realizes f in a *non*-simultaneous manner: it sends a random output of f to the parties in an arbitrary order. When a party gets its part of the output, it instructs the trusted party to either continue sending the output to the other parties, or to send them the abort symbol (i.e., the trusted party "implements" f in a perfect non-simultaneous manner). All notions given in Sections 2.4.2 and 2.4.3 naturally extend to the f-hybrid model, for any functionality f. In addition, the proof of Lemma 2.1 straightforwardly extends to this model. We also make use of the following known fact.

Fact 2.3. *Let f be a polynomial-time computable functionality, and assume there exists a t-party, m-round, α-fair coin-flipping protocol in the f-hybrid model, making at most k calls to f, were t, m, α and k, are function of the security parameter κ. Assuming there exist a protocol for securely computing* OT, *then there exists a t-party, $(O(k \cdot t^2) + m)$-round, $(\alpha + \mathrm{neg}(\kappa))$-fair coin-flipping protocol (in the real world).*

Proof. See in Buchbinder et al. [16].

3 The Many-Party Coin-Flipping Protocol

In Section 3.1, the many-party coin-flipping protocol is defined in an hybrid model. The security of the latter protocol is analyzed in Section 3.2. The (real model) many-party coin-flipping protocol is defined and analyzed in Section 3.3.

3.1 The Hybrid-Model Protocol The protocols below are defined in an hybrid model in which the parties get joint oracle access to several ideal functionalities. We assume the following conventions about the model: all functionalities are guaranteed to function correctly, but do not guarantee fairness: an adversary can abort, and thus preventing the honest parties from getting their output, *after* seeing the outputs of the corrupted parties in its control. We assume identified abort: when a party aborts, its identity is revealed to all other parties. We also assume that when the parties make *parallel* oracle calls, a party that aborts in one of these calls is forced to abort in all of them.

The protocols defined below will not be efficient, even in the hybrid model, since the parties are required to hold presentation of real numbers (which apparently have infinite presentation), we handle this inefficiency when defining the (efficient) real world protocol in Section 3.3.

Protocol $\widehat{\Pi}$ defined next is our final (hybrid model) coin-flipping protocol. This protocol is merely a wrapper for protocol Π: the parties first correlate their private inputs using an oracle to the $\mathsf{Defense}$ functionality, and then interact in Π with these inputs (protocol Π

and the functionality Defense are defined below). For $m, t \in \mathbb{N}$, the t-party, $O(m \cdot t)$-round protocol $\widehat{\Pi}^t_m$ is defined as follows.

PROTOCOL 3.1. ($\widehat{\Pi}^t_m = (\widehat{\mathsf{P}}_1, \ldots, \widehat{\mathsf{P}}_t)$)

Oracle: Defense.

Common input: defense-quality parameter 1^ℓ.

Protocol's description:

1. *Let* $\delta^{\#1}, \ldots, \delta^{\#t}$ *be t-out-of-t shares of* $\frac{1}{2}$.
2. *For every* $\emptyset \neq \mathcal{Z} \subseteq [t]$, *the parties jointly call* $\mathsf{Defense}(1^m, 1^t, 1^\ell, \mathcal{Z}, \delta^{\#1}, \ldots, \delta^{\#t})$, *where* 1^m, 1^t, 1^ℓ, *and* $\mathcal{Z}$ *are common inputs, and input* $\delta^{\#k}$ *is provided by party* P_k. *Let* $\delta^{\#z,\mathcal{Z}}$ *be the output of party* $\widehat{\mathsf{P}}_z$ *returned by this call.*
3. *The parties interact in* $\Pi^t_m = (\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_t)$ *with common input* 1^ℓ. *Party* $\widehat{\mathsf{P}}_z$ *plays the role of* P_z *with private input* $\{\delta^{\#z,\mathcal{Z}}\}_{\emptyset \neq \mathcal{Z} \subseteq [t]}$.

Abort (during step 2): If there is a single remaining party, it outputs an unbiased bit. Otherwise, the remaining parties interact in $\widehat{\Pi}^{t'}_m(1^\ell)$ *for* $t' < t$ *being the number of the remaining parties.*

3.1.1 Protocol Π^r_m

When defining Π^r_m, we make a distinction whether the number of parties is two or larger. We let Π^2_m be the two-party protocol Π^{HT}_m, which is a variant of the of two-party protocol of [30] defined in Section 3.1.5. For the many-party case (more than two), we use the newly defined protocol given below. This distinction between the two-party and many-party cases is made for improving the bias of the final protocol, and all is well-defined if we would have used the protocol below also for the two-party case. On the first read, we encourage the reader to ignore this distinction and assume that the protocol below is also used for the two-party case.

For $m, r \leq t \in \mathbb{N}$, the r-party, $O(m \cdot r)$-round protocol Π^r_m is defined as follows (the functionalities Defense and Coin the protocol uses are defined in Sections 3.1.2 and 3.1.3, respectively).

PROTOCOL 3.2. ($\Pi^r_m = (\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_r)$ (FOR $r > 2$))

Oracles: Defense, *and* Coin.

Common input: defense-quality parameter 1^ℓ.

P_z*'s inputs:* $\{\delta^{\#z,\mathcal{Z}}\}_{\emptyset \neq \mathcal{Z} \subseteq [r]}$.[12]

Protocol's description:

1. *For every* $\emptyset \neq \mathcal{Z} \subsetneq [r]$, *the parties jointly call* $\mathsf{Defense}(1^m, 1^r, 1^\ell, \mathcal{Z}, \delta^{\#1,[r]}, \ldots, \delta^{\#r,[r]})$, *where* $1^m, 1^r, 1^\ell, \mathcal{Z}$ *are common inputs, and input* $\delta^{\#k,[r]}$ *is provided by party* P_k.
   - *For all* $z \in \mathcal{Z}$, *party* P_z *updates* $\delta^{\#z,\mathcal{Z}}$ *to the value it got back from this call.*
2. *Each party* P_z *sends* $\delta^{\#z,[r]}$ *to the other parties.*
   - *All parties set* $\delta = \bigoplus_{z=1}^r \delta^{\#z,[r]}$.
3. *For* $i = 1$ *to* m*:*
   (a) *The parties jointly call* $\mathsf{Coin}(1^m, 1^r, \delta, c_1, \ldots, c_{i-1})$.
      - *For* $z \in \mathcal{Z}$, *let* $(c_i^{\#z}, \delta_i^{\#z})$ *be the output of party* P_z *return by* Coin.

   (b) *For every* $\emptyset \neq \mathcal{Z} \subsetneq [r]$, *the parties jointly call* $\mathsf{Defense}(1^m, 1^r, 1^\ell, \mathcal{Z}, \delta_i^{\#1}, \ldots, \delta_i^{\#r})$, *where* $1^m, 1^r, 1^\ell, \mathcal{Z}$ *are common inputs, and the input* $\delta_i^{\#k}$ *is provided by party* P_k.
      - *For* $z \in \mathcal{Z}$, *party* P_z *updates* $\delta^{\#z,\mathcal{Z}}$ *to the value it got back from this call.*

   (c) *Each party* P_z *sends* $c_i^{\#z}$ *to the other parties.*
      - *All parties set* $c_i = \bigoplus_{z=1}^r c_i^{\#z}$.

Output: All parties output $\mathsf{sign}(\sum_{i=1}^m c_i)$.

Abort: Let $\emptyset \neq \mathcal{Z} \subsetneq [r]$ *be the indices of the remaining parties. If* $\mathcal{Z} = \{z_k\}$, *then the party* P_k *outputs* $\delta^{\#k,\{k\}}$. *Otherwise* ($|\mathcal{Z}| \geq 2$), *assume for ease of notation that* $\mathcal{Z} = [h]$ *for some* $h \in [r-1]$. *To decide on a common output, the parties interact in* $\Pi^h_m = (\mathsf{P}'_1, \ldots, \mathsf{P}'_h)$ *with common input* 1^ℓ, *where party* P_z *plays the role of* P'_z *with private input* $\{\delta^{\#z,\mathcal{Z}'}\}_{\emptyset \neq \mathcal{Z}' \subseteq \mathcal{Z}}$.

That is, at Step 1 the parties use Defense to be instructed what to do if some parties aborts in the reconstruction of the value of δ that happens at Step 2. If Step 1 ends successfully (no aborts), then the expected outcome of the protocol is guaranteed to be δ (even if some parties abort in he reconstruction of δ done in Step 2). If an abort occurs in this Step 1, then the remaining parties use their inputs to interact in a protocol whose expected outcome is δ', for δ' being the input in the call to Defense that generated the parties' input. The key point is that even though δ might be rather far from δ', the corrupted parities who only holds parties information about δ (i.e., the output of Defense), cannot exploit this gap too effectively.

A similar thing happens when flipping each of the coins c_i. The parties first use Coin and Defense to get

[12]The type of $\delta^{\#z,\mathcal{Z}}$ vary according to $|\mathcal{Z}|$. For $|\mathcal{Z}| = 1$, $\delta^{\#z,\mathcal{Z}}$ is simply a $\{0,1\}$ bit, for $|\mathcal{Z}| \geq 2$ it has a more complicated structure, see Section 3.1.3 for details.

shares of the new coin c_i and to get instructed what to do if some parties abort in the reconstruction of c_i. If Step 3b ends successfully, then the expected outcome of the protocol is $\delta_i = \Pr\left[\mathsf{sign}(\sum_{i=1}^{m} c_i) = 1 \mid c_1, \ldots, c_i\right]$, even if some parties abort in the reconstruction of c_i. If an abort occurs in Step 3b, then the remaining parties use their inputs to interact in a protocol whose expected outcome is δ_{i-1}. Also in this case, the corrupted parities cannot exploit the gap between δ_i and δ_{i-1} too effectively.

We note that in the recursive invocations done in the protocol when abort happens, the number of interacting parties in the new protocol is smaller. We also note that all calls to the Defense functionality taken is Step 1 / Step 3b are done in *parallel*. Hence, the resulting protocol has indeed $O(r \cdot m)$ rounds.

Finally, the role of the input parameter ℓ is to optimize the information the calls to Defense leak through the execution of the protocol (including its sub-protocols executions that take place when aborts happen). Recall (see discussion in the introduction) that on one hand, we would like Defense to leak as little information as possible, to prevent an effective attack of the current execution of the protocol. For instance, the value return by Defense in Step 1, should not give too much information about the value of δ. On the other hand, a too hiding Defense will make an interaction done in a sub-protocol, happens if an abort happens, less secure. Parameter ℓ is set to t in the parent call to the protocol done from the t-party protocol $\widehat{\Pi}^t$ and is kept to this value throughout the different sub-protocol executions, enables us to find the optimal balance between these contradicting requirements. See Section 3.1.3 for details.

3.1.2 The Coin Functionality

Functionality Coin performs the (non fair) coin-flipping operation done inside the main loop of Π. It outputs shares of the i-th round's coin c_i, and also shares for the value of expected outcome of the protocol given c_i.[13]

Recall that $\mathcal{B}er(\delta)$ is the Bernoulli probability distribution over $\{0,1\}$ the assigns 1 probability δ, that $\mathcal{C}_\varepsilon$ is the Bernoulli probability distribution over $\{-1,1\}$ that assigns 1 probability $\frac{1}{2}(1+\varepsilon)$, that $\mathcal{C}_{n,\varepsilon}(k) = \Pr\left[\sum_{i=1}^{n} x_i = k\right]$ for x_i's that are i.i.d according to $\mathcal{C}_\varepsilon$, and $\widehat{\mathcal{C}}_{n,\varepsilon}(k) = \Pr_{x \leftarrow \mathcal{C}_{n,\varepsilon}}[x \geq k]$. Also recall that $\widehat{\mathcal{C}}_n^{-1}(\delta)$ is the value $\varepsilon \in [-1,1]$ with $\widehat{\mathcal{C}}_{n,\varepsilon}(0) = \delta$, that $\ell_m(i) = (m+1-i)^2$ (i.e., the number of coins tossed at round i), and that $\mathsf{sum}_m(i) = \sum_{j=i}^{n} \ell_m(j)$ (i.e., the number of coins tossed after round i).

[13]This redundancy in the functionality description, i.e., the shares of coins can be used to compute the the second part of the output, simplifies the presentation of the protocol.

Algorithm 3.1. (Coin)

Input: Parameters 1^m and 1^r, $\delta \in [0,1]$, and coins $c_1, \ldots, c_{i-1}$.

Operation:

```
1. Let $\varepsilon = \widehat{\mathcal{C}}^{-1}_{\mathsf{sum}_m(1)}(\delta)$.
2. Sample $c_i \leftarrow \mathcal{C}_{\ell_m(i),\varepsilon}$.
3. Let $\delta_i = \widehat{\mathcal{C}}_{\mathsf{sum}_m(i+1),\varepsilon}(-\sum_{j=1}^{i} c_j)$
4. Sample $r$ uniform strings $\mathsf{share}^{\#1}, \ldots, \mathsf{share}^{\#r}$ conditioned on $(c_i, \delta_i) = \bigoplus_{i=1}^{r} \mathsf{share}^{\#i}$, and return party $\mathsf{P}_i$ the share $\mathsf{share}^{\#i}$.
```

3.1.3 The Defense Functionality

The Defense functionality is used by protocol Π to "defend" the remaining parties when some corrupted parties abort. When invoked with a subset $\mathcal{Z} \subsetneq [r]$ and $\delta \in [0,1]$, it produces the inputs the parties in $\mathcal{Z}$ need in order to collaborate and produce a δ-biased bit — expected value is δ.

As with protocols Π_m^r, we made a distinction whether $r = 2$ (r is the number of parties that call Defense) or $r > 2$. In the former we use a simple variant of the [30] defense functionality defined in Section 3.1.5, and for all other values we use a newly defined functionality. Also in this case, we encourage the first-time reader to ignore this subtlety, and assume we use the new definition for all cases.

Algorithm 3.2. (Defense functionality for $r > 2$)

Input: Parameters 1^m, 1^r, 1^ℓ, set $\mathcal{Z} \subseteq [r]$ and shares $\{\delta^{\#z}\}_{z \in [r]}$.

Operation: Return $\widetilde{\mathsf{Defense}}(1^m, 1^r, 1^\ell, \mathcal{Z}, \bigoplus_{z \in [r]} \delta^{\#z})$

Namely, Defense just reconstructs δ and calls $\widetilde{\mathsf{Defense}}$ defined below.

Algorithm 3.3. ($\widetilde{\mathsf{Defense}}$)

Input: Parameter 1^m, 1^r, 1^ℓ, set $\mathcal{Z} = \{z_1, \ldots, z_k\} \subsetneq [r]$, and $\delta \in [0,1]$.

Operation:

```
1. If $|\mathcal{Z}| = 1$, let $o_1 \leftarrow \mathcal{C}_\delta$.
2. If $|\mathcal{Z}| = 2$, let $(o_1, o_2) = \mathsf{Defense}^{\mathsf{HT}}(1^m, \delta)$.
3. If $|\mathcal{Z}| > 2$,
    (a) Let $\delta' = \mathsf{Noise}(1^m, 1^\ell, |\mathcal{Z}|, \delta)$.
```

(b) Sample $|\mathcal{Z}|$ uniform shares $o_1, \ldots, o_k$ such that $\delta' = \bigoplus_{i=1}^{k} o_k$.

4. Return o_i to party P_{z_i}, and $\perp$ to the other parties.

It is clear that for the case $|\mathcal{Z}| = 1$, the expected value of the output bit of the party in $\mathcal{Z}$ is indeed δ. Since the expected value of δ' output by $\mathsf{Noise}(\cdot, \delta)$, see below, it δ, it is not hard to see that the same holds also for the case $|\mathcal{Z}| > 2$. Finally, thought somewhat more difficult to verify, the above also holds for the case $|\mathcal{Z}| = 2$ (see Section 3.1.5).

3.1.4 The Noise Functionality

The Noise functionality, invoked by $\widetilde{\mathsf{Defense}}$, takes as input $\delta \in [0,1]$ and returns a "noisy version" of it δ' (i.e., expected value is δ). The amount of noise used is determined by the defense-quality parameter ℓ that reflects the number of players that interact in the parent protocol $\widehat{\Pi}^t$, the number of parties that will use the returned value in their sub-protocol $|\mathcal{Z}|$, and the round complexity of the protocol m.

Definition 3.3. (α-factors) *For $m \geq 1$, $\ell \geq 2$ and $2 \leq k \leq \ell$, let $\alpha(m, \ell, k) = m^{\frac{2^{\ell-3}}{2^{\ell-2}-1} \cdot \frac{2^{k-2}-1}{2^{k-3}}}$.*

Algorithm 3.4. (Noise)

Input: Parameter 1^m, 1^ℓ and 1^k, and $\delta \in [0,1]$.

Operation:

1. Let $\alpha = \alpha(m, \ell, k)$.
2. Let $\varepsilon = \widehat{\mathcal{C}}^{-1}_{\mathsf{sum}_m(1)}(\delta)$.
3. Let $\bar{\mathbf{b}} \leftarrow (\mathcal{C}_\varepsilon)^{\alpha \cdot \mathsf{sum}_m(1)}$.
4. Let δ' be $\Pr_{\mathcal{X} \subseteq [\alpha \cdot \mathsf{sum}_m(1)], |\mathcal{X}| = \mathsf{sum}_m(1)} \left[\sum_{x \in X} \bar{\mathbf{b}}[x] > 0 \right]$.[14]
5. Output δ'.

Namely, Noise sample a vector $\bar{\mathbf{b}}$ of $\alpha \cdot \mathsf{sum}_m(1)$ ε-biased coins. The value of δ' is then determined as the probability to get a positive sum, when sampling $\mathsf{sum}_m(1)$-size subset of coins from $\bar{\mathbf{b}}$.

3.1.5 The Protocol of Haitner and Tsfadia

In this section we define the two-party protocol (the Π^{HT}) and the functionality $\mathsf{Defense}^{\mathsf{HT}}$ we used in the above. The following protocol and functionality are very close variant for those used in Haitner and Tsfadia [30] for construction their three-party coin-flipping protocol. For an elaborated discussion of the ratio underlying the following definitions, see [29].

[14] δ' is the probability that when sampling $\mathsf{sum}_m(1)$ coins from $\bar{\mathbf{b}}$, their sum is positive.

Protocol Π^{HT}. The two-party m-round protocol Π^{HT}_m is defined as follows (the functionality $\mathsf{RoundDefense}^{\mathsf{HT}}$ used by the protocol is defined below). Recall that for $\ell \in \mathbb{N}$, $h(\ell) = \lceil \log \ell \rceil + 1$ is the number of bits it takes to encode an integer in $[-\ell, \ell]$.

Protocol 3.4. ($\Pi^{\mathsf{HT}}_m = (\mathsf{P}_1, \mathsf{P}_2)$)

Common input: round parameter 1^m.

Oracles: $\mathsf{RoundDefense}^{\mathsf{HT}}$.

P_z's input: $\mathbf{c}^{\#\mathbf{z}} \in \{0,1\}^{m \times h(m)}$, $d^z \in \{0,1\}$, and $\mathbf{b}^{\#\mathbf{z},\mathbf{1}}, \mathbf{b}^{\#\mathbf{z},\mathbf{2}} \in \{0,1\}^{2 \cdot \mathsf{sum}_m(1)}$.

Protocol's description:

1. *For $i = 1$ to m:*
   (a) *The parties jointly call $\mathsf{RoundDefense}^{\mathsf{HT}}(1^m, c_1, \ldots, c_{i-1}, \mathbf{c}^{\#\mathbf{1}}[i], \mathbf{c}^{\#\mathbf{2}}[i], \mathbf{b}^{\#\mathbf{1},\mathbf{1}}, \mathbf{b}^{\#\mathbf{1},\mathbf{2}}, \mathbf{b}^{\#\mathbf{2},\mathbf{1}}, \mathbf{b}^{\#\mathbf{2},\mathbf{2}})$, where $(1^m, c_1, \ldots, c_{i-1})$ is the common input, and $(\mathbf{c}^{\#\mathbf{z}}[i], \mathbf{b}^{\#\mathbf{z},\mathbf{1}}, \mathbf{b}^{\#\mathbf{z},\mathbf{2}})$ is provided by the party P_z.*
      - *For all $z \in \{1,2\}$, party P_z updates d^z to the value it got back from this call.*

   (b) *P_1 sends $\mathbf{c}^{\#\mathbf{1}}[i]$ to P_2, and P_2 sends $\mathbf{c}^{\#\mathbf{2}}[i]$ to P_1.*
      - *Both parties set $c_i = \mathbf{c}^{\#\mathbf{1}}[i] \oplus \mathbf{c}^{\#\mathbf{2}}[i]$.*

2. *Both parties output $\mathsf{sign}(\sum_{i=1}^{m} c_i)$.*

Abort: The remaining party P_z outputs d^z.

That is, the parties get correlated shares for the rounds' coins, and they reveal them in the main loop at Step 1b. Prior to revealing them, the parties call the $\mathsf{RoundDefense}^{\mathsf{HT}}$ functionality to get a defense value in case the other party aborts during the coin reconstruction.

Algorithm 3.5. ($\mathsf{RoundDefense}^{\mathsf{HT}}$)

Input: Parameter 1^m, coins $c_1, \ldots, c_{i-1}$, and shares $\mathbf{c}^{\#\mathbf{1}}[i], \mathbf{c}^{\#\mathbf{2}}[i] \in \{0,1\}^{h(m)}$ and $\mathbf{b}^{\#\mathbf{1},\mathbf{1}}, \mathbf{b}^{\#\mathbf{2},\mathbf{1}}, \mathbf{b}^{\#\mathbf{1},\mathbf{2}}, \mathbf{b}^{\#\mathbf{2},\mathbf{2}} \in \{-1,1\}^{2 \cdot \mathsf{sum}_m(1)}$.

Operation:

1. Let $\mathbf{b}^{\mathbf{1}} = \mathbf{b}^{\#\mathbf{1},\mathbf{1}} \oplus \mathbf{b}^{\#\mathbf{2},\mathbf{1}}$, $\mathbf{b}^{\mathbf{2}} = \mathbf{b}^{\#\mathbf{1},\mathbf{2}} \oplus \mathbf{b}^{\#\mathbf{2},\mathbf{2}}$ and $c_i = \mathbf{c}^{\#\mathbf{1}}[i] \oplus \mathbf{c}^{\#\mathbf{2}}[i]$.
2. For both $z \in \{1,2\}$: sample a random $(\mathsf{sum}_m(i+1))$-size subset $\mathcal{W}^z \subset [2 \cdot \mathsf{sum}_m(1)]$, and set d^z to one if $\sum_{j=1}^{i} c_j + \sum_{w \in \mathcal{W}^z} \mathbf{b}^{\mathbf{z}}[w] \geq 0$, and to zero otherwise.
3. Return d^z to party P_z.

Namely, to generate a defense value d_z for $\mathsf{P}z$, $\mathsf{RoundDefense}^{\mathsf{HT}}$ samples $(\mathsf{sum}_m(i+1))$-coins from the vector $\mathbf{b^z}$, adds them to the coin $c_1, \cdots, c_i$ and set d_z to the sign of this sum.

The $\mathsf{Defense}^{\mathsf{HT}}$ functionality. This functionality prepares the inputs for the parties that interact in Π^{HT}. In is invoked by $\widetilde{\mathsf{Defense}}$ when the number of parties to be "defended" is two.

Recall that for $n \in \mathbb{N}$ and $\varepsilon \in [-1,1]$, $\mathcal{C}_{n,\varepsilon}$ is the binomial distribution induced by the sum of n independent random ± 1 coins, taking the value 1 with probability $\frac{1}{2}(1+\varepsilon)$ and -1 otherwise.

Algorithm 3.6. ($\mathsf{Defense}^{\mathsf{HT}}$)

Input: Parameter 1^m and $\delta \in [0,1]$.

Operation:

1. Let $\varepsilon = \widehat{\mathcal{C}}^{-1}_{\mathsf{sum}_m(1)}(\delta)$.
2. For $z \in \{1,2\}$: sample a $(2 \cdot \mathsf{sum}_m(1))$-size vectors $\mathbf{b^z}$ over $\{-1,1\}$, where each element is independently drawn from $\mathcal{C}_\varepsilon$.
3. For $z \in \{1,2\}$: sample a random $(\mathsf{sum}_m(1))$-size subset $\mathcal{I}^z \subset [2 \cdot \mathsf{sum}_m(1)]$, and set d^z to one if $w(\mathbf{b^z}_{\mathcal{I}^z}) \geq 0$, and to zero otherwise.
4. Let $\mathbf{c} = (c_1, \ldots, c_m)$ where for $i \in [m]$, $c_i \leftarrow \mathcal{C}_{\ell_m(i),\varepsilon}$.
5. Sample two uniform shares $\mathbf{c}^{\#\mathbf{1}}, \mathbf{c}^{\#\mathbf{2}}$ such that $\mathbf{c}^{\#\mathbf{1}} \oplus \mathbf{c}^{\#\mathbf{2}} = \mathbf{c}$, and for $z \in \{1,2\}$ two uniform shares $\mathbf{b}^{\#\mathbf{1},\mathbf{z}}, \mathbf{b}^{\#\mathbf{2},\mathbf{z}}$ such that $\mathbf{b}^{\#\mathbf{1},\mathbf{z}} \oplus \mathbf{b}^{\#\mathbf{2},\mathbf{z}} = \mathbf{b^z}$.
6. Return $((\mathbf{c}^{\#\mathbf{1}}, \mathbf{b}^{\#\mathbf{1},\mathbf{1}}, \mathbf{b}^{\#\mathbf{1},\mathbf{2}}, d^1), (\mathbf{c}^{\#\mathbf{2}}, \mathbf{b}^{\#\mathbf{2},\mathbf{1}}, \mathbf{b}^{\#\mathbf{2},\mathbf{2}}, d^2))$.

Namely, on input δ as input, $\mathsf{Defense}^{\mathsf{HT}}$ calculates a corresponding ε at Step 1. Then, $\mathsf{Defense}^{\mathsf{HT}}$ uses this ε to sample the rounds' coins c_i, and the banks that are used by $\mathsf{RoundDefense}^{\mathsf{HT}}$ to give a defense value in every round of the loop of Π^{HT}_m.

3.2 Security Analysis of the Hybrid-Model Protocol

In this section we prove that the protocol cannot be biased too much by a fail-stop adversary. Here is our main theorem in the case of a fail-stop adversary:

Theorem 3.1. *For $m \equiv 1 \bmod 12$, $t = t(m) \in \mathbb{N}$, such that $t = o(\log m)$, protocol $\widehat{\Pi}^t_m(1^t)$ is a $(t \cdot m)$-round, t-party, $O\left(\frac{t \cdot 2^t \cdot \sqrt{\log m}}{m^{1/2+1/(2^{t-1}-2)}}\right)$-fair, coin-flipping protocol, against unbounded fail-stop adversaries, in the $(\mathsf{Defense}, \mathsf{Coin})$-hybrid model.*

We prove Theorem 3.1 in Section 3.2.4, but before that we introduce the main tools and concept used for this proof. Leakage from two-step boolean process used to bound attack in Step 1 is presented in Section 3.2.1. Binomial games used to bound an attack inside the loop of Protocol 3.2 are introduced in Section 3.2.2. Finally, in Section 3.2.3 we state several simple claims about the protocol.

3.2.1 Leakage from Two-Step Boolean Processes

Our main tool for analyzing the effect of an abort in Step 1 of protocol Π^r_m, when $r > 2$, is bounding the leakage from the relevant "two-step boolean process". A two-step boolean process is a pair of jointly-distributed random variables (A,B), where B is over $\{0,1\}$ and A is over an arbitrary domain $\mathcal{A}$. It is instructive to think that the process' first step is choosing A, and its second step is to choose B as a random function of A. A leakage function f for a two-step process (A,B) is simply a randomized function over the support of A. We will be interested in bounding by how much the expected outcome of B changes when $f(A)$ leaks. This change is captured via the notion of prediction advantage.

Definition 3.5. (prediction advantage) *For a two-step process $\mathsf{P} = (A,B)$ and a leakage function f for P, define the* prediction advantage $\Gamma_{\mathsf{P},f}$ *by* $\Gamma_{\mathsf{P},f}(h) = |\Pr[B=1] - \Pr[B=1 \mid f(A)=h]|$.

We now define the notions of an hypergeometric process, and of vector leakage function. As we shall see later on, the boolean process that induced in Item 1 of protocol Π^r_m, can be viewed as such a hypergeometric process, coupled to a vector leakage function.

Definition 3.6. (vector leakage function) *Let s, α be integers. A randomized function f is a* (s,α)-vector leakage function *for the two-step Boolean process (A,B), if on input $a \in \mathrm{Supp}(A)$, it outputs a vector in $\{-1,1\}^{\alpha \cdot s}$ according to $(\mathcal{C}_\varepsilon)^{\alpha \cdot s}$, for $\varepsilon = \widehat{\mathcal{C}}^{-1}_s(\mathrm{E}[B \mid A=a])$.*

Definition 3.7. (Hypergeometric process) *Let $s, \beta \in \mathbb{N}$ and $\delta \in [0,1]$. An* (s,β,δ)-hypergeometric process *is the two-step Boolean process (A,B) defined by*

1. *$A = \widehat{\mathcal{HG}}_{\beta \cdot s, w(v), s}(0)$, for $v \leftarrow (\mathcal{C}_\varepsilon)^{\beta \cdot s}$, for $\varepsilon = \widehat{\mathcal{C}}^{-1}_s(\delta)$.*
2. *$B \leftarrow \mathcal{B}er(A)$,*

In Section 3.2.4 we use the following lemma to bound the gain an adversary can achieve by aborting at Step 1. The proof can be found in Buchbinder et al. [16].

Lemma 3.1. *Assume $s, \alpha, \beta \in \mathbb{N}$ and $\delta \in [0,1]$, satisfy*

1. $2 \le \alpha < \beta \le s$,
2. $\frac{\alpha+\sqrt{s}}{s} \cdot \log^2 s \le 10^{-5} \cdot \sqrt{\frac{\alpha}{\beta}}$, *and*
3. $\sqrt{\frac{\alpha}{\beta}} \cdot \log s \le \frac{1}{100}$.

Let $\mathsf{P} = (A, B)$ be a (s, β, δ)-hypergeometric process, and let f be an (s, α)-vector leakage function for P. Then, there exists some universal constant $\lambda > 0$ such that

$$\Pr_{h \leftarrow f(A)}\left[\Gamma_{\mathsf{P},f}(h) > \lambda \cdot \sqrt{\log s} \cdot \frac{\sqrt{\alpha}}{\beta}\right] \le \frac{1}{s^2}.$$

3.2.2 Online-Binomial Games Our main tool for analyzing the effect an abort in the main loop of the protocol has, is bounding the bias of the relevant "online-binomial games". Following the informal discussion given in Section 1, we give here a formal definition of online binomial game. While in the introduction we referred to a very narrow notion of binomial game, here we cover a wider class of games, by letting the challenger to toss more than one coin each round (so in some sense this is a weighted binomial game), and the coins it toss not necessary need to be unbiased.

Definition 3.8. (online-binomial game) *Let $m \in \mathbb{N}$, $\varepsilon \in [-1,1]$, and let $\{C_1, \ldots, C_m\}$, be independent random variables with $C_i \leftarrow \mathcal{C}_{(m-i+1)^2,\varepsilon}$. Let f be a randomized function over $[m] \times \mathbb{Z} \times \mathbb{Z}$. The m-round* online binomial game *$\mathsf{G}_{m,\varepsilon,f}$ is defined to be $\mathsf{G}_{m,\varepsilon,f} = \{C_1, \ldots, C_m, f\}$. We refer to each C_i as the i'th* round coins, *and to f as the* hint *function.*

We will be interested in bounding by how much can boas the outcome of such a game.

Definition 3.9. (The bias of $G_{m,\varepsilon,f}$) *Let $\mathsf{G} = \mathsf{G}_{m,\varepsilon,f} = \{C_1, \ldots, C_m, f\}$ be an m-round online binomial game, and let $\mathcal{H}$ be the range of f. For $i \in \{0, \ldots, m\}$, let $S_i = \sum_{j=1}^{i} C_i$, letting $S_0 = 0$, and let $H_i = f(i, S_{i-1}, C_i)$. For $b \in \mathbb{Z}$, and $i \in [m+1]$, let $\delta_i(b) = \Pr[S_m \ge 0 \mid S_{i-1} = b]$, and for $h \in \mathcal{H}$ let $\delta_i(b,h) = \Pr[S_m \ge 0 \mid S_{i-1} = b,\ H_i = h]$.*

For an algorithm B, let I be the first round in which B outputs 1 in the following m-round process: in round i, algorithm B is getting input (S_{i-1}, H_i) and outputs a $\{0,1\}$-value. Let $I = m+1$ if B never outputs a one. The bias B gains in G *is defined by*

$$\mathsf{Bias}_{\mathsf{B}}(\mathsf{G}) = \left| \mathrm{E}\left[\delta_I(S_{I-1}) - \delta_I(S_{I-1}, H_I)\right] \right|$$

The bias of G *is defined by $\mathsf{Bias}_{m,\varepsilon,f} = \mathsf{Bias}(\mathsf{G}) = \max_{\mathsf{B}}\{\mathsf{Bias}_{\mathsf{B}}(\mathsf{G})\}$, where the maximum is over* all *possible algorithms B.*

Namely, in the i'th round the algorithm B is getting the sum of the coins flipped up to previous round - S_{i-1}, and a "hint" $H_i = f(i, S_{i-1}, C_i)$. If the strategy decides to abort, it get rewarded by $|\delta_i(S_{i-1}, H_i) - \delta_i(S_{i-1})|$. Hence, a strategy "goal" is to find the round in which the above gain is maximized.[15]

In the proof of Theorem 3.1, we use the following lemma (proven in Buchbinder et al. [16]).

Definition 3.10. (Vector hint) *For $m, \ell \in \mathbb{N}$ and $\varepsilon \in [-1,1]$, define the random function $f^{\mathsf{vec}}_{m,\varepsilon,\ell} \colon [m] \times \mathbb{Z} \times \mathbb{Z} \mapsto \{-1,1\}^\ell$ as follows: on input (i, b, c), it calculates $\delta = \widehat{\mathcal{C}}_{\mathsf{sum}_m(i+1),\varepsilon}(-b-c)$, and $\varepsilon := \widehat{\mathcal{C}}^{-1}_{\mathsf{sum}_m(1)}(\delta)$, and returns a random sample from $(\mathcal{C}_\varepsilon)^\ell$.*

Lemma 3.2. *For $m \in \mathbb{N}$, $k \in [m]$, $\varepsilon \in [-1,1]$, and $f = f^{\mathsf{vec}}_{m,\varepsilon,k \cdot \mathsf{sum}_m(1)}$ for some, and let G be the binomial game $\mathsf{G}_{m,\varepsilon,f}$ according to Definition 3.8. Then $\mathsf{Bias}_{\mathsf{G}} \in O(\frac{\sqrt{k}}{m} \cdot \sqrt{\log m})$.*

3.2.3 Basic Observations about Protocol 3.1 The following simple facts are used within the proof of Theorem 3.1. We start with a simple fact regarding the outcome of the $\mathsf{Defense}$ functionality.

Fact 3.1. *Let $m \ge 1$, $r > 2$, $\ell \in \mathbb{N}$, $\bar{\delta} \in [0,1]$ and $\mathcal{Z} = (z_1, \ldots, z_{|\mathcal{Z}|}) \subseteq [r]$, and let $S = (S_1, \ldots, S_r) = \mathsf{Defense}(1^m, 1^r, 1^\ell, \mathcal{Z}, \bar{\delta})$. Let $\mathsf{outcome}(S)$ be the outcome of a non-aborting execution of protocol $\Pi_m^{|\mathcal{Z}|}$ on common input 1^ℓ, and the j'th party private input is set to S_{z_j}. Then for every $\mathcal{B} \subset [r]$ with $\mathcal{Z} \not\subseteq \mathcal{B}$ and for every $s \in \mathrm{Supp}(S^{\mathcal{B}} = \{S_z\}_{z \in \mathcal{B}})$, it holds that $\mathrm{E}\left[\mathsf{outcome}(S) \mid S^{\mathcal{B}} = s\right] = \bar{\delta}$.*

Namely, in an honest interaction that follows an abort, the expected outcome of the interaction is $\bar{\delta}$, for $\bar{\delta}$ being the input in the last call to $\mathsf{Defense}$ that happened before the abort. The latter holds, even conditioned on the partial information held by the corrupted parties.

Proof. Assume without loss of generality that $\mathcal{B} = \{2, \ldots, r\}$, and that $\mathcal{Z} = \{1, \ldots, |\mathcal{Z}|\}$. Consider an honest execution of protocol $\Pi_m^{|\mathcal{Z}|}$, when parties in $\mathcal{Z}$

[15] The choice of the gain function in the above definition is somewhat artificial, and the "right" choice is to set the gain function to $\max_{\mathsf{B}}\{|\mathsf{Bias}_{m,\varepsilon,f}(\mathsf{B})|\}$. Yet, for for out application the above simpler variant suffices.

start with private inputs S_j for $1 \le j \le |\mathcal{Z}|$[16], and fix some $s = \langle s_2, \ldots, s_r \rangle \in \mathrm{Supp}(S^{\mathcal{B}})$. By construction of $\mathsf{Defense}$ functionality, and especially the fact that it breaks the output into random shares, it holds that $\mathrm{E}\left[\bigoplus_{i=1}^{|\mathcal{Z}|} S_i \mid S^{\mathcal{B}} = s\right] = \bar{\delta}$. Writing it a bit differently:

$$\underset{S_1}{\mathrm{E}}\left[S_1 \oplus s_2 \oplus \ldots \oplus s_{|\mathcal{Z}|}\right] = \bar{\delta} \tag{3.1}$$

By construction of Protocol 3.2 it holds that

$$\mathrm{E}\left[\mathsf{outcome}(S) \mid S_1, \ldots, S_{|\mathcal{Z}|}\right] = S_1 \oplus \ldots \oplus S_{|\mathcal{Z}|} \tag{3.2}$$

Putting it together we get:

$$\begin{aligned}
&\mathrm{E}\left[\mathsf{outcome}(S) \mid S^{\mathcal{B}} = s\right] \\
&= \mathrm{E}\left[\mathsf{outcome}(S) \mid S_2 = s_2, \ldots, S_r = s_r\right] \qquad (3.3)\\
&= \underset{S_1}{\mathrm{E}}\left[\mathrm{E}\left[\mathsf{outcome}(S) \mid S_1, S_2 = s_2, \ldots, S_r = s_r\right]\right] \\
&= \underset{S_1}{\mathrm{E}}\left[S_1 \oplus s_2 \oplus \ldots \oplus s_{|\mathcal{Z}|}\right] \\
&= \bar{\delta}
\end{aligned}$$

as required.

We remind the reader that the α-factors are: $\alpha(m, \ell, k) = m^{\frac{2^{\ell-3}}{2^{\ell-2}-1} \cdot \frac{2^{k-2}-1}{2^{k-3}}}$ (see Definition 3.3). The following fact states some basic properties of the α-factors.

Fact 3.2. *Let $m \ge 1$ and $\ell \ge 2$ be two integers, and denote for simplicity $\alpha_k = \alpha(m, \ell, k)$. It holds that*

1. $\alpha_{\ell-1} = m^{1 - \frac{1}{2^{\ell-2}-1}}$.
2. $\alpha_2 = 1$.
3. $\frac{\sqrt{\alpha_2}}{\alpha_3} = \ldots = \frac{\sqrt{\alpha_{\ell-3}}}{\alpha_{\ell-2}} = \frac{\sqrt{\alpha_{\ell-2}}}{\alpha_{\ell-1}} = \frac{\sqrt{\alpha_{\ell-1}}}{m} = \frac{1}{m^{\frac{1}{2} + \frac{1}{2^{\ell-1}-2}}}$.

Proof. Immediate by definition.

3.2.4 Proving Theorem 3.1

Proof. [Proof of Theorem 3.1]

By construction, in an all-honest execution the parties output a uniform bit, so it left to prove that the protocol cannot be biased by too much. Let A be a fail-stop adversary controlling the parties $\{\widehat{\mathsf{P}}_z\}_{z \in \mathcal{C}}$ for some $\mathcal{C} \subsetneq [t]$. Throughout, we assume without loss of generality that if a an adversary instructs a corrupted party to abort at a given round, it does so *after* seeing the honest parties' messages of that round.

Let V be the (joint) view of the parties controlled by A, let V_i is the prefix of V at the end of round i, and V_i^- be the prefix of V_i with the i'th round abort messages (if any) removed. Let $\mathsf{val}(v)$ be the expected outcome of an honest (non-aborting) execution of the protocol by the parties that do not abort in v, conditioned on v (see Definition 2.5).

For $k \in [t-1]$, let I_k be the k-th aborting round (that is, the k'th round in which at least one party aborts). Let $I_k = \perp$ if less than k aborting rounds happen, and let $V_\perp = V_\perp^-$. By Lemma 2.1, to prove the theorem it sufficient to show that:

$$\left|\underset{V}{\mathrm{E}}\left[\sum_{k=1}^{t-1} \mathsf{val}(V_{I_k}) - \mathsf{val}(V_{I_k}^-)\right]\right| \le O\left(\frac{t \cdot 2^t \cdot \sqrt{\log m}}{m^{1/2 + 1/(2^{t-1}-2)}}\right). \tag{3.4}$$

Since

$$\begin{aligned}
&\left|\underset{V}{\mathrm{E}}\left[\sum_{k=1}^{t-1} \mathsf{val}(V_{I_k}) - \mathsf{val}(V_{I_k}^-)\right]\right| \\
&= \left|\sum_{k=1}^{t-1} \underset{V}{\mathrm{E}}\left[\mathsf{val}(V_{I_k}) - \mathsf{val}(V_{I_k}^-)\right]\right| \\
&\le \sum_{k=1}^{t-1} \left|\underset{V}{\mathrm{E}}\left[\mathsf{val}(V_{I_k}) - \mathsf{val}(V_{I_k}^-)\right]\right|,
\end{aligned}$$

to prove the theorem we show that

$$\left|\underset{V}{\mathrm{E}}\left[\mathsf{val}(V_{I_k}) - \mathsf{val}(V_{I_k}^-)\right]\right| \le O\left(\frac{2^t \cdot \sqrt{\log m}}{m^{1/2 + 1/(2^{t-1}-2)}}\right) \tag{3.5}$$

for every $1 \le k \le t-1$.

So fix some k, $1 \le k \le t-1$. The k'th abort can occur in one of the following places:

- In Step 2 of the parent protocol $\widehat{\Pi}_m^t$ (can only happen if $k = 1$).
- During the execution of protocol Π_m^r, for some $r \le t$.

Since by construction aborting in Step 2 gives nothing to the adversary, it is left to prove that Equation (3.5) holds for aborting done during an execution of Π_m^r.

Let $R = R(k)$ be the number of active parties when the k'th abort occur (that means that it occurs during the execution of Π_m^R). We show that for any value of r, it holds that

$$\left|\underset{V|R=r}{\mathrm{E}}\left[\mathsf{val}(V_{I_k}) - \mathsf{val}(V_{I_k}^-)\right]\right| \le O\left(\frac{2^t \cdot \sqrt{\log m}}{m^{1/2 + 1/(2^{t-1}-2)}}\right) \tag{3.6}$$

[16] The parties that participate in this execution have more inputs for the case that some of them will abort later on. Since we are interested in an honest execution we can ignore those additional inputs.

and Equation (3.5) will follow.

Let $I = I_k$. In the following we assume that $r > 2$, the proof for $r = 2$ is very similar to the proof given in Haitner and Tsfadia [29] using an improved bound on the bias of online-binomial games, and can be found in the full version of this paper [16].

We assume without loss of generality that $I \neq \perp$ (i.e., an abort occurs) and thus I is a round of the interaction in protocol Π^r_m. Let $\mathcal{Z} \subset [r]$ be the indices of the remaining non-aborting parties, let Δ be the value of δ calculated in Step 2, and let $\bar{\Delta}$ be the $\delta \in [0,1]$ parameter passed to the last call to $\widetilde{\mathsf{Defense}}$ before I (note that I, Δ and $\bar{\Delta}$ are random variables). By Fact 3.1, it holds that

$$\mathsf{val}(V_I) = \bar{\Delta} \tag{3.7}$$

We first prove that if the aborting round I is in Step 2, Step 3a, or Step 3c, then the adversary gains no bias *at all*. That is, conditioned that the I is in one of these steps, then

$$\mathsf{val}(V_I) = \mathsf{val}(V_I^-)$$

Start with I being in Step 2. By definition, $\mathsf{val}(V_I^-) = \bar{\Delta}$, which by Equation (3.7) is exactly $\mathsf{val}(V_I)$. The case that I is in Step 3c, follows by a analogous argument. For the case that I is in Step 3a, note that since c_i and δ_i are shared using an r-out-of-r secret sharing schemes, V_I contains no information about c_i and δ_i, and thus $\mathsf{val}(V_I^-) = \mathsf{val}(V_{I-1})$. If I is the very first round to reach Step 3a, then by construction $\mathsf{val}(V_{I-1}) = \bar{\Delta}$. If I is not the first round to reach Step 3a, then by definition $\mathsf{val}(V_{I-1}) = \Pr\left[\mathsf{sign}(\sum_{i=1}^m c_i) = 1 \mid V_{I-1}\right]$, which by construction is also equal to $\bar{\Delta}$. Equation (3.7) yields that $\mathsf{val}(V_I) = \bar{\Delta}$, and we conclude that $\mathsf{val}(V_I) = \mathsf{val}(V_I^-)$ also in this case.

In the rest of the proof we separately handle the cases that I is in Step 1 or Step 3b of the protocol.

The case I is in Step 1. Let τ denote the round index when the execution of protocol $\widehat{\Pi}^t_m$ reaches Step 1 of the embedded protocol Π^r_m. Let $v' \in \operatorname{Supp}(V_{\tau-1})$, and let N be the messages that the corrupted parties receive during round τ. We assume without loss of generality that N contains also the vectors of coins sampled in Step 3 of functionality Noise, and was used to generate the defense value for the corrupted parties; obviously, an adversary that can bias the protocol without this additional information, can be emulated by an adversary that get this additional information. Let $\bar{\delta}$ be the value of the δ parameter that was given as parameter to the last call of $\widetilde{\mathsf{Defense}}$ in the execution (as described in v').[17] Note that v' contains the value of $\bar{\delta}$.

In the following we condition on $V_{\tau-1} = v'$. We prove Equation (3.6) for the case I is in Step 1, by showing that:

$$\left|\mathop{\mathrm{E}}_{V_\tau \mid I=\tau}\left[\mathsf{val}(V_I) - \mathsf{val}(V_I^-)\right]\right| \cdot \Pr\left[I = \tau\right] \leq O\left(\frac{2^r \cdot \sqrt{\log m}}{m^{1/2+1/(2^{\ell-1}-2)}}\right).$$

The proof follow by the next claim (proven below).

Claim 3.11. *It holds that*

$$\Pr_{n \leftarrow N}\left[\bar{\delta} - \Pr\left[\Delta = 1 \mid N = n\right] > \lambda \frac{\sqrt{2^r \alpha_{r-1}}}{\alpha_r}\sqrt{\log m}\right] \leq \frac{1}{m^2}$$

Namely, with high probability, after the adversary sees the messages of round τ, the value of Δ is not far from $\bar{\delta}$. It follows that

$$\begin{aligned}
&\mathop{\mathrm{E}}_{V_\tau \mid I=\tau}\left[\mathsf{val}(V_I) - \mathsf{val}(V_I^-)\right] \\
&= \mathop{\mathrm{E}}_{V_\tau \mid I=\tau}\left[\mathsf{val}(V_I)\right] - \mathop{\mathrm{E}}_{V_\tau \mid I=\tau}\left[\mathsf{val}(V_I^-)\right] && (3.8)\\
&= \bar{\delta} - \mathop{\mathrm{E}}_{V_\tau \mid I=\tau}\left[\mathsf{val}(V_I^-)\right] && (3.9)\\
&= \bar{\delta} - \mathop{\mathrm{E}}_{V_\tau \mid I=\tau}\left[\mathrm{E}\left[\Delta \mid V_\tau\right]\right] && (3.10)\\
&= \mathop{\mathrm{E}}_{V_\tau \mid I=\tau}\left[\bar{\delta} - \mathrm{E}\left[\Delta \mid V_\tau\right]\right] \\
&= \mathop{\mathrm{E}}_{N \mid I=\tau}\left[\bar{\delta} - \mathrm{E}\left[\Delta \mid N\right]\right] && (3.11)\\
&= \mathop{\mathrm{E}}_{N \mid I=\tau}\left[\bar{\delta} - \Pr\left[\Delta = 1 \mid N\right]\right].
\end{aligned}$$

Equation (3.9) holds since Fact 3.1 yields that $\mathrm{E}_{V_\tau \mid I=\tau}\left[\mathsf{val}(V_I)\right] = \bar{\delta}$, Equation (3.10) holds since $\mathsf{val}(V_I^-) = \mathrm{E}\left[\Delta \mid V_\tau\right]$ by the definition of $\mathsf{val}(\cdot)$, and Equation (3.11) holds because V_τ is $V_{\tau-1}$ concatenate with N, and we condition on $V_{\tau-1} = v'$.

By triangle inequality, it holds that

$$\left|\mathop{\mathrm{E}}_{V_\tau \mid I=\tau}\left[\mathsf{val}(V_I) - \mathsf{val}(V_I^-)\right]\right| \cdot \Pr\left[I = \tau\right] \leq \mathop{\mathrm{E}}_{N \mid I=\tau}\left[\left|\bar{\delta} - \Pr\left[\Delta = 1 \mid N\right]\right|\right] \cdot \Pr\left[I = \tau\right]$$

Since

$$\begin{aligned}
&\mathop{\mathrm{E}}_{N \mid I=\tau}\left[\left|\bar{\delta} - \Pr\left[\Delta = 1 \mid N\right]\right|\right] \cdot \Pr\left[I = \tau\right] \\
&\leq \mathop{\mathrm{E}}_{N}\left[\left|\bar{\delta} - \Pr\left[\Delta = 1 \mid N\right]\right|\right] \\
&= \sum_{n \in \operatorname{Supp}(N)} \left|\bar{\delta} - \Pr\left[\Delta = 1 \mid N = n\right]\right| \cdot \Pr\left[N = n\right],
\end{aligned}$$

[17] By construction, such a call always exists.

Claim 3.11 yields that

$$\begin{aligned}&\mathop{\mathrm{E}}_{N\mid I=\tau}\left[\left|\overline{\delta}-\Pr\left[\Delta=1\mid N\right]\right|\right]\cdot\Pr\left[I=\tau\right]\\ (3.12)\qquad &\leq 1\cdot\frac{1}{m^2}+\lambda\cdot\frac{\sqrt{2^r\cdot\alpha_{r-1}}}{\alpha_r}\cdot\sqrt{\log m}\end{aligned}$$

for some universal constant λ. Finally, since $\frac{\sqrt{2^r\cdot\alpha_{r-1}}}{\alpha_r}\leq\frac{\sqrt{2^r\cdot\alpha_{\ell-1}}}{m}=\frac{\sqrt{2^r}}{m^{\frac{1}{2}+\frac{1}{2^{\ell-1}-2}}}$ (Fact 3.2), we conclude that

$$\begin{aligned}&\left|\mathop{\mathrm{E}}_{V_\tau\mid I=\tau}\left[\mathsf{val}(V_I)-\mathsf{val}(V_I^-)\right]\right|\cdot\Pr\left[I=\tau\right]\\ &\leq O\left(\frac{2^r\cdot\sqrt{\log m}}{m^{1/2+1/(2^{\ell-1}-2)}}\right),\end{aligned}$$

as required.

The case I is in Step 3b. The crux of the proof lies in the following claim, proved below, that relates the maximal bias achieved by an adversary that only aborts at round Step 3b and the (maximal) bias of a certain type of binomial game (see Section 3.2.2).

Claim 3.12. *Let $f=f^{\mathsf{vec}}_{m,\varepsilon,2^r\cdot\alpha_{r-1}\cdot\mathsf{sum}_m(1)}$ be according to Definition 3.10, let $\mathsf{G}=\mathsf{G}_{m,\varepsilon,f}$ be a binomial game with vector hint, according to Definition 3.8 and* Bias *be according to Definition 3.9. Assuming* A *only aborts at Step 3b, then* $\mathrm{E}\left[\left|\mathsf{val}(V_I)-\mathsf{val}(V_I^-)\right|\right]\leq\mathsf{Bias}(\mathsf{G})$.

The proof of the theorem now follows by the above claim and Lemma 3.2.

Proving Claim 3.11

Proof. [Proof of Claim 3.11] Define the two-step process (see Section 3.2.1 for an introduction about leakage from two-step boolean processes) $P=(A,B)$, for $A=\Delta$, and $B=\mathcal{B}er(A)$, and define a leakage function f for P by $f(a)=N|_{A=a}$ (i.e., the messages received by the corrupted parties at round τ). By definition, $\overline{\delta}-\Pr\left[\Delta=1\mid N=n\right]=\Gamma_{P,f}(m)$ for every $n\in\mathrm{Supp}(N)$. Hence, it is left to prove that

$$(3.13)\qquad \Pr_{n\leftarrow N}\left[\Gamma_{P,f}(n)>\lambda\cdot\frac{\sqrt{2^r\cdot\alpha_{r-1}}}{\alpha_r}\cdot\sqrt{\log m}\right]\leq\frac{1}{m^2}.$$

Let $P'=(A',B')$ be a $\langle\mathsf{sum}_m(1),\alpha_r,\overline{\delta}\rangle$-hypergeometric process (see Definition 3.7), and let f' be a $(\mathsf{sum}_m(1),2^r\cdot\alpha_{r-1})$-vector leakage function (see Definition 3.6) for P'. By construction, it holds that $P\equiv P'$. We remind the reader that we assume that N contains also the vectors of coins sampled at Step 3 in the Noise algorithm, and that the messages that the corrupted parties get are a random function of those vectors. Hence, for every $a\in\mathrm{Supp}(A)$, $f(a)$ is actually $f'(a)$ and some random function of $f(a)$. Thus, for proving Equation (3.13) it suffices to show that [18]

$$(3.14)\qquad \Pr_{h\leftarrow f'(A')}\left[\Gamma_{P',f'}(h)>\lambda\cdot\frac{\sqrt{2^r\cdot\alpha_{r-1}}}{\alpha_r}\cdot\sqrt{\log m}\right]\leq\frac{1}{m^2}$$

We prove the above equation by applying Lemma 3.1 for the hypergeometric process (A',B') with the vector leakage function f', and parameters $s=\mathsf{sum}_m(1)$, $\beta=\alpha_r$ and $\alpha=2^r\cdot\alpha_{r-1}$. Note that the first and third conditions of Lemma 3.1 trivially holds for this choice of parameters, whereas the second condition holds since $\frac{\log^2 s}{\sqrt{s}}=o(\sqrt{\frac{\alpha}{\beta}})$ and since $\frac{\alpha}{s}\cdot\log^2 s=o(\sqrt{\frac{\alpha}{\beta}})$ for $r\leq t=o(\log m)$. Therefore, Lemma 3.1 yields that

$$\Pr_{h\leftarrow f'(A')}\left[\left|\Gamma_{P',f'}(h)\right|>\lambda'\sqrt{\log s}\cdot\frac{\sqrt{\alpha}}{\beta}\right]\leq\frac{1}{s^2},$$

for some universal constant $\lambda'>0$. We conclude that

$$\begin{aligned}&\Pr_{h\leftarrow f'(A')}\left[\left|\Gamma_{P',f'}(h)\right|>2\lambda'\sqrt{\log m}\cdot\frac{\sqrt{2^r\cdot\alpha_{r-1}}}{\alpha_r}\right]\\ &\leq\frac{1}{s^2}\leq\frac{1}{m^2},\end{aligned}$$

and the proof of the claim follows.

Proving Claim 3.12.

Proof. [Proof of Claim 3.12] Assuming A achieves bias α, we construct a strategy B for G that achieves the same bias α. It will follow that $\mathrm{E}\left[\left|\mathsf{val}(V_I)-\mathsf{val}(V_I^-)\right|\right]$, conditioned on I being at Step 3b, is at most $\mathsf{Bias}(\mathsf{G})$.

We define algorithm B for G as follows.

Algorithm 3.7. (algorithm B)

Operation:

1. Start emulating a random execution of protocol $\widehat{\Pi}^t_m(1^\ell)$, with A controlling parties $\mathsf{P}_1,\ldots,\mathsf{P}_{t-1}$, until the main loop of Π^r_m is reached. If not reached, algorithm B never aborts. Set $\delta=\bigoplus_{z=1}^r\mathsf{share}^{\#z,[r]}$, where those shares are taken from the private inputs of the parties participating in Π^r_m.
2. For $i=1$ to m:
   (a) Let (s_{i-1},h_i) be the i'th message sent by the challenger.

[18] We rely here on a claim given in Buchbinder et al. [16] that says that additional information which is a random function of the given hint, does not improve the prediction advantage.

(b) If $i > 1$, emulate Step 3c: let $c_{i-1} = s_i - s_{i-1}$ and set $c_{i-1}^{\#r}$ such that $c_{i-1} = \bigoplus_{z\in[r]} c_{i-1}^{\#z}$. Emulate the reconstruction of c_{i-1}, letting $c_{i-1}^{\#r}$ be the message of the honest party.

(c) Emulate Step 3a: send the corrupted parties $2 \cdot (r-1)$ random strings $(c_i^{\#1}, \delta_i^{\#1}, \ldots, c_i^{\#r-1}, \delta_i^{\#r-1})$ as the answers of Coin.

(d) Emulate Step 3b: emulate the parallel calls to $\mathsf{Defense}$ using the hint h_i.

Recall that the $\mathsf{Defense}$ functionality is merely a deterministic wrapper for the $\widetilde{\mathsf{Defense}}$ functionality, and the latter, in turn, is a wrapper to the Noise functionality. Hence, it suffices to shows how to use h_i for emulating these calls to Noise. The Noise functionality uses $\alpha_{r-1} \cdot \mathsf{sum}_m(1)$ independent $\mathcal{C}_\varepsilon$-biased coins per call, and there are at most 2^r such calls. Also note that hint h_i is a vector of $2^r \cdot \alpha_{r-1} \cdot \mathsf{sum}_m(1)$ entries of independent from $\mathcal{C}_\varepsilon$.

To emulate this step, the samples in h_i for these samples needed by Noise.

- If A aborts at this step, output 1 (i.e., abort at round i). Otherwise, output 0 (i.e., continue to next round).

By construction, A's view in the above emulation has the same distribution as in his view in the execution of Protocol 3.1. Recall that the bias of B for a binomial game $\mathsf{G} = \mathsf{G}_{m,\varepsilon,f}$ is defined by $\mathsf{Bias}_\mathsf{B}(\mathsf{G}) = \mathrm{E}\left[|\delta_I(S_{I-1}) - \delta_I(S_{I-1}, H_I)|\right]$, where I is the aborting round of B, S_j is the sum of coins tossed up to round j, $\delta_i(s_{i-1})$ is the expected outcome of the binomial game given $S_{I-1} = s_{i-1}$, and $\delta_i(s_{i-1}, h_i)$ is the expected outcome of the binomial game given $S_{I-1} = s_{i-1}$ and the hint in round i is h_i. By construction, $\mathsf{val}(V_I) = \delta_I(S_{I-1})$ and $\mathsf{val}(V_I^-) = \delta_I(S_{I-1}, H_I)$. It follows that $\mathsf{Bias}_{m,\varepsilon,f}(\mathsf{B}) = \mathrm{E}\left[\left|\mathsf{val}(V_I) - \mathsf{val}(V_I^-)\right|\right]$. Since $\mathsf{Bias}(\mathsf{G}) = \max_\mathsf{B}\{\mathsf{Bias}_\mathsf{B}(\mathsf{G})\}$, we conclude that $\mathrm{E}\left[\left|\mathsf{val}(V_I) - \mathsf{val}(V_I^-)\right|\right] \leq \mathsf{Bias}(\mathsf{G})$.

3.3 Proving Main Theorem

In this section we prove our main result, an $O(m)$-round, t-party coin-flipping protocol, in the real (non-hybrid) model, that is $O(\frac{t^3 \cdot 2^t \cdot \sqrt{\log m}}{m^{1/2+1/(2^{t-1}-2)}})$-fair.

Theorem 3.2. (Main theorem) *Assuming protocols for securely computing* OT *exist, then for any polynomially bounded, polynomial-time computable, integer functions $m = m(\kappa)$ and $t = t(\kappa) \in o(\log m)$, there exists a t-party, m-round, $O(\frac{t^3 \cdot 2^t \cdot \sqrt{\log m}}{m^{1/2+1/(2^{t-1}-2)}})$-fair, coin-flipping protocol.*

Proof. [Proof of Theorem 3.2] We compile our hybrid protocol defined in Section 3.1 into the desired real-world protocol. The main part of the proof is showing how to modify for any $m', t' \in \mathbb{N}$, the $O(m't')$-round, t'-party hybrid protocol $\widehat{\Pi}_{m'}^{t'}$ (see Protocol 3.1), into a form that allows this compilation This change involves several steps all using standard techniques. In the following we fix m' and t' and let $\widehat{\Pi} = \widehat{\Pi}_{m'}^{t'}$.

The first change is that $\widehat{\Pi}$ (through protocol Π) uses real numbers. Specifically, the parties keep the value of the expected outcome δ, which is a real number in $[0,1]$, and also keep shares for such values of δ. We note that the value of δ is always set to the probability that when sampling some k ε-biased $\{-1,1\}$-coins, the bias is at least $b \in \mathbb{Z}$. Where in turn, ε is the value such that the sum of n ε-biased coins, is positive with probability δ', for some δ' whose value is already held by the parties. It follows that δ has short description given the value of δ' (i.e., the values of k and b), and thus all δ have short descriptions.

The second change is to modify the functionalities used by the protocol as oracles into ones that are polynomial-time computable in m' and t', without hurting the security of the protocol. By inspection, the only calculation that need to be treated is the calculations done in Step 1 of Coin, Step 2 in Noise, and Step 1 in $\mathsf{Defense}^{\mathsf{HT}}$. To be concrete, we focus on the calculation of $\varepsilon = \widehat{\mathcal{C}}_{\mathsf{sum}_m(1)}^{-1}(\delta)$ for some $\delta \in [0,1]$ done in Coin. Via sampling, for any $p \in \mathrm{poly}$ one can efficiently estimate ε by a value $\widetilde{\varepsilon}$ such that $|\varepsilon - \widetilde{\varepsilon}| \leq \frac{1}{p(m)}$ with save but negligible probability in m. Since ε is merely used for sampling $q(m) \in \mathrm{poly}$ ε-bias $\{-1,1\}$ coins, it follows that statistical distance between the parties' views in random execution of $\widehat{\Pi}_m^t$ and the efficient variant of $\widehat{\Pi}_m^t$ that uses the above estimation, is at most $\frac{q(m)}{p(m)} + \mathrm{neg}(m)$ which is in $O(1/m)$ for large enough p. It follows that Theorem 3.1 also holds with respect to the above efficient implementation of Coin.

Our next change is to make all the oracles call made by the parties to be sequential (i.e., one after the other). To do that we replace the parallel calls to $\mathsf{Defense}$ done in Step 1 and Step 3b of Protocol 3.2, with a single call per step. This done by modifying $\mathsf{Defense}$ to get as input the inputs provided by the parties for all parallel calls, compute the answer of each of this calls, and return the answers in an aggregated manner to the parties. Since our hybrid model dictates that a single abort in one of the parallel calls to $\mathsf{Defense}$ aborts all calls, it is clear that this change does not effect the correctness and security of protocol $\widehat{\Pi}$.

Our last change it to make the protocol secure against arbitrary adversaries (not only fail-stop ones). Using information-theoretic one-time message authentication codes (cf., [40]), the functionalities Coin, $\mathsf{Defense}$ and protocol $\widehat{\Pi}$ can be compiled into functionalities and protocol that maintain the same correctness, essentially the same efficiency, and the resulting protocol is $(\gamma+\operatorname{neg}(m',t'))$-fair against *arbitrary* adversaries, assuming the protocol γ-fair against fail-stop adversaries.

We are finally ready to define the real-model coin-flipping protocol. Let $\widetilde{m} = \widetilde{m}(\kappa) = \lceil m(\kappa)/c \cdot t(\kappa)^3 \rceil - a$, for $c > 0$ to be determined by the analysts, and $a \in \{0,\ldots,11\}$ is the value such that $\widetilde{m}(\kappa) - a \equiv 1 \bmod 12$.[19] We consider the $O(t\cdot\widetilde{m})$-round, t-party, polynomial-time hybrid-model protocol $\widetilde{\Pi}$, that on input κ, the parties act as in $\widehat{\Pi}^t_{\widetilde{m}}(1^t)$.

By the above observations and since, by assumption, $t = t(\kappa) \in o(\log m)$, assuming protocols for securely computing OT exist, Fact 2.3 yields that there exists an $O(t^3\widetilde{m} + t\cdot\widetilde{m})$-round, t-party, polynomial-time protocol $\breve{\Pi}$ correct coin-flipping protocol, that is $(\gamma(\kappa)+\operatorname{neg}(\kappa))$-fair in the *standard model*, assuming that protocol $\widetilde{\Pi}$ is $\gamma(\kappa)$-fair in the $(\mathsf{Defense},\mathsf{Coin})$-hybrid model. By choosing c in the definition of $\widetilde{m}$ large enough, we have that yields that protocol has (at most) m rounds. Theorem 3.1 and the above observations yields the $\widetilde{\Pi}$ is a $\left(O\left(\frac{t\cdot 2^t\cdot\sqrt{\log\widetilde{m}}}{\widetilde{m}^{1/2+1/(2^{t-1}-2)}}\right) = O\left(\frac{t^2\cdot 2^t\cdot\sqrt{\log m}}{m^{1/2+1/(2^{t-1}-2)}}\right)\right)$-fair in the $(\mathsf{Defense},\mathsf{Coin})$-hybrid model, yielding that $\breve{\Pi}$ is $O\left(\frac{t^3\cdot 2^t\cdot\sqrt{\log m}}{m^{1/2+1/(2^{t-1}-2)}}\right)$-fair.

References

[1] D. Aharonov, A. Ta-Shma, U. Vazirani, and A. C. Yao. Quantum bit escrow. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2000.

[2] W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In *Advances in Cryptology – EUROCRYPT 2001*, 2001.

[3] B. Alon and E. Omri. Almost-optimally fair multiparty coin-tossing with nearly three-quarters malicious. Cryptology ePrint Archive, Report 2016/800, 2016. http://eprint.iacr.org/2016/800.

[4] N. Alon and M. Naor. Coin-flipping games immune against linear-sized coalitions. *SIAM Journal on Computing*, pages 46–54, 1993.

[5] A. Ambainis. A new protocol and lower bounds for quantum coin flipping. *J. Comput. Syst. Sci.*, 68 (2):398–416, 2004.

[6] A. Ambainis, H. Buhrman, Y. Dodis, and H. Röhrig. Multiparty quantum coin flipping. In *Proceedings of the 18th Annual IEEE Conference on Computational Complexity*, pages 250–259, 2004.

[7] G. Asharov. Towards characterizing complete fairness in secure two-party computation. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 291–316, 2014. doi: 10.1007/978-3-642-54242-8_13.

[8] 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.

[9] B. Awerbuch, M. Blum, B. Chor, S. Goldwasser, and S. Micali. How to implement brachas o (log n) byzantine agreement algorithm. 1985.

[10] A. Beimel, E. Omri, and I. Orlov. Protocols for multiparty coin toss with dishonest majority. In *Advances in Cryptology – CRYPTO 2010*, volume 6223, pages 538–557, 2010.

[11] A. Beimel, Y. Lindell, E. Omri, and I. Orlov. $1/p$-secure multiparty computation without honest majority and the best of both worlds. In *Advances in Cryptology – CRYPTO 2011*, pages 277–296, 2011.

[12] M. Ben-Or and N. Linial. Collective coin flipping. *ADVCR: Advances in Computing Research*, 5, 1989.

[13] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, 1988.

[14] I. Berman, I. Haitner, and A. Tentes. Coin flipping of any constant bias implies one-way functions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 817–836, 2014.

[19] Note that the total number of coins, which equals to $\frac{\widetilde{m}(\widetilde{m}+1)(2\widetilde{m}+1)}{6}$, is odd for $\widetilde{m} \equiv 1 \bmod 12$.

[15] M. Blum. How to exchange (secret) keys. *ACM Transactions on Computer Systems*, 1983.

[16] N. Buchbinder, I. Haitner, N. Levi, and E. Tsfadia. Fair coin flipping: Tighter analysis and the many-party case. www.cs.tau.ac.il/~iftachh/papers/3PartyCF/ManyPartyCF_Full.pdf. Manuscript. Preliminary version in *SODA '17*.

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

[18] R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 364–369, 1986.

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

[20] D. Dachman-Soled, Y. Lindell, M. Mahmoody, and T. Malkin. On the black-box complexity of optimally-fair coin tossing. In *tcc11*, pages 450–467, 2011.

[21] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[22] U. Feige. Noncryptographic selection protocols. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, 1999.

[23] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 197–206, 2008.

[24] O. Goldreich. *Foundations of Cryptography – VOLUME 2: Basic Applications*. Cambridge University Press, 2004.

[25] S. D. Gordon and J. Katz. Partial fairness in secure two-party computation. In *Advances in Cryptology – EUROCRYPT 2011*, pages 157–176, 2010.

[26] S. D. Gordon, C. Hazay, J. Katz, and Y. Lindell. Complete fairness in secure two-party computation. *Journal of the ACM*, 58(6):24, 2011.

[27] I. Haitner. Implementing oblivious transfer using collection of dense trapdoor permutations. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004*, pages 394–409, 2004.

[28] 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.

[29] I. Haitner and E. Tsfadia. An almost-optimally fair three-party coin-flipping protocol. www.cs.tau.ac.il/~iftachh/papers/3PartyCF/QuasiOptimalCF_Full.pdf. Manuscript. Preliminary version in *STOC '14*.

[30] I. Haitner and E. Tsfadia. An almost-optimally fair three-party coin-flipping protocol. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 817–836, 2014.

[31] 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*, pages 1153–1218, 2009.

[32] I. Haitner, O. Reingold, S. Vadhan, and H. Wee. Inaccessible entropy. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 611–620, 2009.

[33] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, pages 1364–1396, 1999.

[34] W. Hoeffding. Probability inequalities for sums of bounded random variables, 1963.

[35] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 230–235, 1989.

[36] Y. Kalai. Smooth projective hashing and two-message oblivious transfer. In *Advances in Cryptology – EUROCRYPT 2005*, 2005.

[37] J. Katz. On achieving the "best of both worlds" in secure multiparty computation. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 11–20, 2007.

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

[39] T. Moran and M. Naor. Basing cryptographic protocols on tamper-evident seals. In *ICALP: Annual International Colloquium on Automata, Languages and Programming*, 2005.

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

[41] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, pages 151–158, 1991.

[42] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, pages 151–158, 1991.

[43] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.

[44] A. Russell and D. Zuckerman. Perfect information leader election in log* n + 0 (1) rounds. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS)*, 1999.

[45] M. Saks. A robust noncryptographic protocol for collective coin flipping. *SIJDM: SIAM Journal on Discrete Mathematics*, 2, 1989.

[46] M. Scala. Hypergeometric tail inequalities: ending the insanity, 2009.