# Protocols for Multiparty Coin Toss with a Dishonest Majority*

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

Eran Omri‡
Department of Computer Science and Mathematics
Ariel University
Ariel, Israel

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

September 3, 2013

**Abstract**

Coin-tossing protocols are protocols that generate a random bit with uniform distribution, although some corrupted parties might try to bias the output. These protocols are used as a building block in many cryptographic protocols. Cleve [STOC 1986] has shown that if at least half of the parties can be corrupted, then, in any $r$-round coin-tossing protocol, the corrupted parties can cause a bias of $\Omega(1/r)$ to the bit that the honest parties output. However, for more than two decades the best known protocols had bias $t/\sqrt{r}$, where $t$ is the number of corrupted parties. Recently, in a surprising result, Moran, Naor, and Segev[TCC 2009] constructed an $r$-round two-party coin-tossing protocol with the optimal bias of $O(1/r)$. We extend the results of Moran et al. to the *multiparty model* where fewer than 2/3 of the parties are corrupted. The bias of our protocol is proportional to $1/r$ and doubly exponential in the gap between the number of corrupted parties and the number of honest parties in the protocol. In particular, for a constant number of parties, where fewer than 2/3 of them are corrupted. we present an $r$-round $m$-party coin-tossing protocol with an optimal bias of $O(1/r)$. Furthermore, we achieve the same bias even when the number of parties $m$ is non-constant and the number of corrupted parties is $m/2 + O(1)$.

**Keywords.** Fair Coin Tossing, Multiparty Computation, Dishonest Majority, Secure with Abort; Cheat Detection.

*A preliminary version of this work appeared in *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 538–557. Springer-Verlag, 2010.

†Supported by ISF grant 938/09.

‡Part of the research was done while the author was a post-doctoral fellow at Bar-Ilan University, supported by the European Research Council as part of the ERC project "LAST". Part of the research was done while the author was a post-doctoral fellow at Ben-Gurion University, supported by ISF grant 860/06.

§Supported by ISF grant 938/09 and by the Frankel Center for Computer Science.

# Contents


**1 Introduction** **2**
- 1.1 Our Results . . . 3
- 1.2 The Main Ideas Underlying Our Protocols . . . 4
- 1.3 Previous and Follow-up Works . . . 4

**2 Preliminaries** **5**
- 2.1 The Real vs. Ideal Paradigm . . . 6
  - 2.1.1 $1/p$-Indistinguishability and $1/p$-Secure Computation . . . 7
- 2.2 Security with Abort and Cheat Detection . . . 8
- 2.3 Cryptographic Tools . . . 9
- 2.4 The Two-Party Protocol of Moran et al. . . . 10

**3 Coin Tossing with a Dishonest Majority – A Warm-Up** **11**
- 3.1 Multiparty Coin Tossing When Half of the Parties are Corrupted . . . 11
- 3.2 A 5-Party Protocol that Tolerates up to 3 Corrupted Parties . . . 13
  - 3.2.1 Eliminating the Dealer for the 5-Party Protocol . . . 14

**4 Our Main Construction** **17**

**5 A Protocol with an On-Line Dealer** **18**
- 5.1 The Simulator for the On-Line Dealer Protocol . . . 19
- 5.2 Proof of the Correctness of the Simulation . . . 21

**6 Omitting the On-Line Dealer** **29**
- 6.1 The Protocol Without the Dealer . . . 29
  - 6.1.1 Proving the Feasibility of the Reconstruction . . . 30
- 6.2 The Simulator for the Protocol Without a Dealer . . . 33
- 6.3 Proof of the Correctness of the Simulation . . . 36
  - 6.3.1 The Initialization Step . . . 37
  - 6.3.2 The Interaction Rounds of the Protocol . . . 38
  - 6.3.3 The Termination Step . . . 39
  - 6.3.4 Concluding the Proof of Theorem 6 . . . 41

**7 Coin-Tossing Protocol for any Constant Fraction of Corrupted Parties** **41**

**References** **45**

**A On Achieving Cheat Detection** **46**

# 1 Introduction

Secure multiparty computation in the corrupted model allows distrustful parties to compute a function securely (e.g., compute it correctly and privately). Designing such secure protocols is a delicate task with a lot of subtleties. An interesting and basic functionality for secure computation is coin tossing – generating a random bit with uniform distribution. This is a simple task where the parties have no inputs. However, already this task raises questions of fairness and how corrupted parties can bias the output. Understanding what can be achieved for coin tossing in various settings can be considered as a step towards understanding general secure and fair multiparty computation. Indeed, some of the early works on secure computation were on coin tossing, e.g., [5, 2, 8]. Furthermore, coin tossing and its variants are used as a basic tool in constructing many protocols, e.g., the protocol of [12]. Secure protocols for coin tossing are a digital analogue of physical coin tossing, which have been used throughout history to resolve disputes.

The main problem in designing coin-tossing protocols is the prevention of a bias of the output. The bias of a coin-tossing protocol measures the maximum influence of the adversary controlling a subset of corrupted parties on the output of the honest parties, where the bias is $0$ if the output is always uniformly distributed and the bias is $1/2$ if the adversary can force the output to be always (say) 1. To demonstrate the problems of designing a coin-tossing protocol, we describe Blum's two-party coin-tossing protocol [5].

**Example 1.1** (Blum's coin-tossing protocol [5])**.** *To toss a coin, Alice and Bob execute the following protocol.*

- Alice chooses a random bit $a$ and sends a commitment $c = \mathrm{commit}(a)$ to Bob.
- Bob chooses a random bit $b$ and sends it to Alice.
- Alice sends the bit $a$ to Bob together with $\text{de-commit}(\mathrm{c})$.
- If Bob does not abort during the protocol, Alice outputs $a \oplus b$, otherwise she outputs a random bit.
- If Alice does not abort during the protocol and $c$ is a commitment to $a$, then Bob outputs $a \oplus b$, otherwise he outputs a random bit.

*If Alice is corrupted, then she can bias the output toward (say) 1. If $a \oplus b = 1$, she opens the commitment and Bob outputs 1. However, if $a \oplus b = 0$, Alice aborts, and Bob outputs 1 with probability $1/2$. Altogether, the probability that the honest Bob outputs 1 is $3/4$. It can be proved that this is the best that Alice can do in this protocol, and, hence, the bias of the protocol is $1/4$. This protocol demonstrates the problems caused by parties aborting the protocol and the need to define how the output of the other parties is computed after such aborts.*

While the above protocol is a significant improvement over naive protocols whose bias is $1/2$, the protocol still has a constant bias. If *more than half* of the parties are honest, then, using general secure multiparty protocols, there are constant-round protocols with negligible bias (assuming a broadcast channel), e.g., the protocol of [24]. Cleve [8] proved that when at least half of the parties can be corrupted, the bias of every protocol with $r$ rounds is $\Omega(1/r)$. In particular, this proves that without honest majority no protocol with polynomial number of rounds (in the security parameter) can have negligible bias. On the positive side, it was shown in [2, 8] that there is a two-party protocol with bias $O(1/\sqrt{r})$. This can be generalized to an $m$-party protocol that tolerates any number of corrupted parties and has bias $O(t/\sqrt{r})$, where $t$ is a bound on the number of corrupted parties. Further details on the protocols of [2, 8] appear in Section 7. All the above mentioned protocols and the protocols we describe below are secure against computationally-bounded adversary and assume suitable cryptographic assumptions. Cleve and Impagliazzo [9] have shown that, in a weaker model compared to the model that we assume, where commitments are available only as

black-box (and no other assumptions are made), the bias of every coin-tossing protocol is $\Omega(1/\sqrt{r})$.[1] The protocols of [5, 2, 8] are in this model.

The question of whether or not there exists a coin-tossing protocol without an honest majority that has bias $O(1/r)$ was open for many years. Recently, in a breakthrough result, Moran, Naor, and Segev [22] constructed an $r$-round two-party coin-tossing protocol with bias $O(1/r)$. Moran et al. ask the following question:

> "An interesting problem is to identify the optimal trade-off between the number of parties, the round complexity, and the bias. Unfortunately, it seems that several natural variations of our approach fail to extend to the case of more than two parties."

## 1.1 Our Results

Our main contribution is a multi-party coin-tossing protocol that has small bias when fewer than $2/3$ of the parties are corrupted.

**Theorem 1** (Informal)**.** *Let $n$ be the security parameter, and $m = m(n), t = t(n)$, and $r = r(n)$ be such that $m/2 \leq t < 2m/3$. There exists an $r$-round $m$-party coin-tossing protocol tolerating $t$ corrupted parties that has bias $O(2^{2^{k+1}}/r')$, where $k = 2t - m$ and $r' = r - O(k+1)$.*

The above protocol nearly has the desired dependency on $r$, i.e., the dependency implied by the lower bound of Cleve [8]. However, its dependency on $k$ (where $k$ is the difference between the number of corrupted parties and the number of honest parties) has, in general, a prohibitive cost. In addition, observe that the above protocol is meaningful only when $r' > 2^{2^{k+1}}$, hence, the number of rounds and communication complexity of the obtained protocol is at least double exponential in $k$. Nevertheless, there are interesting cases where the bias is $O(1/r)$.

**Corollary 2** (Informal)**.** *Let $m$ and $t$ be* constants *(in the security parameter) such that $m/2 \leq t < 2m/3$, and let $r = r(n)$ be an integer. There exists an $r$-round $m$-party coin-tossing protocol tolerating $t$ corrupted parties that has bias $O(1/r)$.*

For example, we construct an $r$-round 5-party coin-tossing protocol tolerating 3 corrupted parties that has bias $8/(r - O(1))$ (this follows from our general construction in Sections 4–6).

Notice that the protocol of Theorem 1 depends on $k$ and not on the number of corrupted parties $t$. Thus, it is efficient when $k$ is small compared to $m$ and $n$.

**Corollary 3** (Informal)**.** *Let $n$ be the security parameter, and $m = m(n), t = t(n)$, and $r = r(n)$ be such that $t = m/2 + O(1)$. There exists an $r$-round $m$-party coin-tossing protocol tolerating $t$ corrupted parties that has bias $O(1/r)$.*

For example, for any even $m$ we construct an $r$-round $m$-party coin-tossing protocol tolerating $m/2$ corrupted parties that has bias $1/(2r - O(1))$. Furthermore, even when $t = 0.5m + 0.5 \log\log m - 1$, the bias of our protocol is small, namely, $O(m/(r - O(\log\log m)))$.

We also reduce the bias compared to previous protocols when at least 2/3 of the parties are corrupted. The bias of the $m$-party protocol of [2, 8] is $O(t/\sqrt{r})$. We present in Section 7 a protocol whose bias is $O(1/\sqrt{r})$ when $t/m$ is constant, that is, when the fraction of corrupted parties is constant we get rid of the factor $t$ in the bias.

[1] The lower-bound of [9] holds in a stronger model that we will not discuss in this paper.

We consider a communication model where all parties can only communicate through an authenticated broadcast channel. We assume a synchronous model; however, the adversary is rushing. We note that our results can be translated to a model with authenticated point-to-point channels with a PKI (in an $m$-party protocol, the translation will increase the number of rounds by a multiplicative factor of $O(m)$). Thus, our results hold in the two most common models for secure multiparty computation.

### 1.2 The Main Ideas Underlying Our Protocols

We generalize the two-party protocol of Moran et al. [22] (abbreviated as MNS protocol) to the multi-party setting. In the MNS protocol, in each round Alice and Bob obtain bits that are their output if the other party aborts: If a party aborts in round $i$, then the other party outputs the bit it obtained in round $i-1$. Furthermore, there is a special round $i^*$; prior to round $i^*$ the bits obtained by Alice and Bob are random independent bits and from round $i^*$ onward the bits obtained by Alice and Bob are the same fixed bit. The adversary can bias the output only if it guesses $i^*$.

In our protocol, in each round there are many bits that are used to compute the output when various subsets abort. We define a collection of subsets of the parties and each subset obtains a bit. The bits are chosen similarly to the MNS protocol: prior to $i^*$ they are uniform and independent; from $i^*$ onward they are fixed. In our case we cannot give the bits themselves to the parties (as in the MNS protocol). We rather use a few layers of secret-sharing schemes to allow the parties to compute each of these bits only at a particular point in time. For every subset in the collection, we use a first secret-sharing scheme to share the bit of the subset among the parties of the subset. We use an additional secret-sharing scheme to share the shares of the first secret-sharing scheme. The threshold in the latter secret sharing scheme is chosen such that the protocol can proceeds until enough parties aborted. In the round when the number of aborted parties ensures that there is an honest majority, an appropriate subset in the collection is chosen, its bit is reconstructed, and this bit is the output of the honest parties. The description of how to implement these ideas appears in Sections 4–6.

The construction of the MNS protocol is presented in two phases. In the first phase they present a protocol with a trusted dealer, for which an adversary can only inflict bias of $O(1/r)$. Then, they show how to implement this protocol in the real world, using a constant round secure-with-abort multiparty protocol, as well as secret-sharing and authentication schemes. This can be seen as a general transformation from any two-party coin-tossing protocol with a trusted dealer, into a real world two-party coin-tossing protocol. We observe that the transformation of Moran et al. to a real-world protocol requires some further care for the multiparty case generalization. We show how this can be achieved by adopting the definition of secure multiparty computation of [1], which requires the protocol to detect a cheating party: At the end of the protocol either the honest parties hold a correct output or all honest parties agree on a party (or parties) that cheated during the protocol.

### 1.3 Previous and Follow-up Works

*$1/p$-secure protocols.* Coin-tossing is an example of $1/p$-secure computation. Informally, a protocol is $1/p$-secure if every adversary cannot harm the protocol with probability greater than $1/p$ (e.g., it cannot bias the output with probability greater than $1/p$). The formal definition of $1/p$-secure computation appears in Definition 2.3.

Katz [20] presented a simplified version of $1/p$-security called $1/p$-security *with abort*, which is, roughly speaking, a $1/p$-secure computation in which the adversary is allowed to abort in the ideal world of the

execution. Katz [20] presented a protocol, for any functionality $\mathcal{F}$, with full security when there is an honest majority, as well as $1/p$-security with abort for any number of corrupted parties.

Gordon and Katz [17] defined $1/p$-security and constructed 2-party $1/p$-secure protocols for every functionality whose size of either the domain or the range of the functionality is polynomial (in the security parameter). They also show an impossibility result when both the domain and range are super-polynomial.

In a follow-up work, Beimel et al. [4] study multiparty $1/p$-secure protocols for general functionalities. The main result in [4] is constructions of $1/p$-secure protocols that are resilient against *any* number of corrupted parties provided that the number of parties is constant and the size of the range of the functionality is at most polynomial in the security parameter $n$. The protocols in [4] combine ideas from the protocols presented in this paper and those presented in [20]. In addition, the bias of the protocol of [4] is $O(1/\sqrt{r})$.

*Multiparty protocols for coin-tossing in the preprocessing model.* Ishai et al. [19] constructed a new type of secret-sharing schemes in which the reconstruction algorithm informs each honest player of the correct set of cheaters, i.e., each honest player gets a list of all the parties that input a wrong share in the reconstruction phase. This scheme can be used to convert our multiparty coin-tossing protocols into *unconditional* multiparty coin-tossing protocols in the *preprocessing model*, namely, assuming that there exists an off-line dealer which computes some values in a preprocessing phase, gives them to the parties according to the secure-with-abort model, and halts. Such a transformation eliminates the cryptographic assumptions while the bias of the resulting protocols is not effected. In addition, the impossibility result of Cleve [8] applies to the preprocessing model as well.

**Organization.** In Section 2, we provide the security definitions of multi-party protocols, a description of the cryptographic tools used in this paper, and a brief overview of the MNS protocol. In Section 3, we present two warm-up constructions for multiparty coin-tossing with bias $O(1/r)$ where $r$ is the number of rounds in the protocol. The first construction considers the case that at most half of the parties are corrupted (however, there is no majority of honest parties). The second construction solves the problem of coin tossing with 5 parties, where at most 3 are corrupted. In Sections 4–6 we present our main result – a coin-tossing protocol that has nearly optimal bias and can tolerate up to $2/3$ fraction of corrupted parties. In Section 7 we describe an $r$-round $m$-party coin-tossing protocol that tolerates up to $t$ dishonest parties, where $t$ is some constant fraction of $m$.

## 2 Preliminaries

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

In this work we consider a malicious static computationally-bounded adversary, i.e., a non-uniform algorithm (Turing machine) that runs in a polynomial-time. The adversary is allowed to corrupt some subset of the parties. That is, before the beginning of the protocol, the adversary corrupts a subset of the parties that may arbitrarily deviate from the protocol, and thereafter the adversary sees the messages sent to the corrupted parties and controls the messages sent by the corrupted parties. The honest parties follow the instructions of the protocol.

The parties communicate in a synchronous network, using only a broadcast channel. On one hand, if a party broadcasts a message, then all other parties see *the same* message. This ensures some consistency between the information the parties have. On the other hand, there are no private channels and all the parties see all the messages. The adversary is rushing, that is, in each round the adversary hears the messages sent by the honest parties before broadcasting the messages of the corrupted parties for this round (thus, the messages broadcast by corrupted parties can depend on the messages of the honest parties broadcast in this round). Such an adversary is the more realistic model as perfect synchronicity is hard to achieve.

## 2.1 The Real vs. Ideal Paradigm

The security of multiparty computation protocols is defined using the real vs. ideal paradigm [11, 6]. In this paradigm, we consider the real-world model, in which protocols are executed. We then formulate an ideal model for executing the task at hand. This ideal model involves a trusted party whose functionality captures the security requirements of the task. Finally, we show that the real-world protocol "emulates" the ideal-world protocol: For any real-life adversary $\mathcal{A}$ there should exist an ideal-model adversary $\mathcal{S}$ (also called simulator) such that the global output of an execution of the protocol with $\mathcal{A}$ in the real-world model is distributed similarly to the global output of running $\mathcal{S}$ in the ideal model. In the coin-tossing protocol, the parties do not have inputs. Thus, to simplify the definitions, we define secure computation without inputs (except for the security parameters).

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

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

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

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

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

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

In this work we consider a few formulations of the ideal world (e.g., full security and secure-with-abort), and consider composition of a few protocols, all being executed in the same real world, however, each secure with respect to a different ideal world. We prove the security of the resulting protocol, using the hybrid model techniques of Canetti [6].

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

As explained in the introduction, the ideal functionality $\mathrm{CoinToss}()$ cannot be implemented when there is no honest majority. We use $1/p$-secure computation, defined by [16, 20], to capture the divergence from the ideal world. This notion applies to general secure computation. We start with some notation.

A function $\mu(\cdot)$ is *negligible* if for every positive polynomial $q(\cdot)$ and all sufficiently large $n$ it holds that $\mu(n) < 1/q(n)$. A *distribution ensemble* $X = \{X_{a,n}\}_{a\in\{0,1\}^*,n\in\mathbb{N}}$ is an infinite sequence of random variables indexed by $a \in \{0,1\}^*$ and $n \in \mathbb{N}$.

**Definition 2.1** (Statistical Distance and $1/p$-indistinguishability)**.** *We define the* statistical distance *between two random variables $A$ and $B$ as the function*

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

*For a function $p(n)$, two distribution ensembles $X = \{X_{a,n}\}_{a\in\{0,1\}^*,n\in\mathbb{N}}$ and $Y = \{Y_{a,n}\}_{a\in\{0,1\}^*,n\in\mathbb{N}}$ are* computationally $1/p$-indistinguishable*, denoted $X \overset{1/p}{\approx} Y$, if for every non-uniform polynomial-time algorithm $D$ there exists a negligible function $\mu(\cdot)$ such that for every $n$ and every $a \in \{0,1\}^*$,*

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

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

We next define the notion of $1/p$-secure computation [17, 20, 4]. The definition uses the standard real/ideal paradigm [11, 6], except that we consider a completely fair ideal model (as typically considered in the setting of honest majority), and require only $1/p$-indistinguishability rather than indistinguishability.

**Definition 2.2** (Perfect $1/p$-secure computation)**.** *An $m$-party protocol $\Pi$ is said to* perfectly $1/p$*-secure compute a functionality $\mathcal{F}$ if for every adversary $\mathcal{A}$ in the real model, there exists a polynomial-time adversary $\mathcal{S}$ in the ideal model such that for every $n \in \mathbb{N}$ and for every* $\mathrm{aux} \in \{0,1\}^*$

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

**Definition 2.3** ($1/p$-secure computation [17, 20, 4])**.** *Let $p = p(n)$ be a function. An $m$-party protocol $\Pi$ is said to $1/p$-securely compute a functionality $\mathcal{F}$ if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ in the real model, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model corrupting the same parties as $\mathcal{A}$ such that the following two distribution ensembles are computationally $1/p(n)$-indistinguishable*

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

**Remark 2.4.** *The definition of $1/p$-secure computation allows agreement to fail with probability $1/p$. In the coin-tossing functionality all (honest) parties should agree on the same bit. Our protocols achieve a slightly stronger security requirement than $1/p$-security, where the honest parties disagree on the bit only with a negligible probability (however, the bit might be biased).*

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

**Definition 2.5** (Secure computation)**.** *An $m$-party protocol $\Pi$ is said to* securely compute *a functionality $\mathcal{F}$, if for every $c \in N$, the protocol $\Pi$ is $(1/n^c)$-securely compute the functionality $\mathcal{F}$.*

**Definition 2.6.** *We say that a protocol is a coin-tossing protocol with bias $1/p$ if it is a $1/p$-secure protocol for the functionality* $\mathrm{CoinToss}()$.

## 2.2 Security with Abort and Cheat Detection

We present here a definition of secure multiparty computation that is more stringent than standard definitions of secure computation with abort. This definition extends the definition for secure computation as given by Aumann and Lindell [1]. Roughly speaking, our definition requires that one of two events is possible: If at least one party deviates from the prescribed protocol, then the adversary obtains the outputs of these parties (but nothing else), and all honest parties are notified by the protocol that these parties have aborted. Otherwise, the protocol terminates normally, and all parties receive their outputs. Again, we consider the restricted case where parties hold no private inputs. The formal definition uses the real vs. ideal paradigm as discussed in Section 2.1. We next describe the appropriate ideal model.

**Execution in the ideal model.** Let $D \subseteq [m]$ denote the indices of the corrupted parties, controlled by an adversary $\mathcal{A}$. An ideal execution proceeds as follows:

**Inputs:** Each party obtains a security parameter $1^n$. The adversary $\mathcal{A}$ receives an auxiliary input denoted $\mathrm{aux}$.

**Trusted party sends outputs to adversary:** The trusted party computes $\mathcal{F}(1^n)$ with uniformly random coins computes and sends the appropriate outputs to all parties $p_j$ such that $j \in D$.

**Adversary instructs the trusted party to continue or halt:** $\mathcal{A}$ sends either a "`continue`" message or $\{\text{"}\texttt{cheat}_j\text{"}\}_{j \in J}$ to the trusted party on behalf of a set of corrupted parties indexed by $J$. If it sends a "`continue`" message, the trusted party sends the appropriate output to the parties $p_j$ for $j \notin D$ (i.e., to all honest parties). Otherwise, the trusted party sends the set $J$ to all parties $p_j$ for $j \notin D$.

**Outputs:** An honest party always outputs the message it obtained from the trusted party. The corrupted parties output nothing. The adversary $\mathcal{A}$ outputs an probabilistic polynomial-time computable function of the auxiliary input $\mathrm{aux}$, and the outputs obtained from the trusted party.

We let $\mathrm{IDEAL}^{\mathrm{CD}}_{\mathcal{F},\mathcal{S}(\mathrm{aux})}(1^n)$ and $\mathrm{REAL}_{\Pi,A(\mathrm{aux})}(1^n)$ be defined as in Section 2.1 (where in this case $\mathrm{IDEAL}^{\mathrm{CD}}_{\mathcal{F},\mathcal{S}(\mathrm{aux})}(1^n)$ refers to the above execution with cheat detection of $\mathcal{F}$). This ideal model is different from the ideal world without cheat detection (see, e.g., [11]); without cheat detection, the trusted party sends a $\perp$ symbol when the trusted party gets an abort message. With cheat detection, the honest parties *know* the identities of the corrupted parties that cause the cheat. This cheat detection is achieved by most multiparty protocols, including that of [12], but not all (e.g., the protocol of [13] does not meet this requirement). Using this notation we define secure computation with abort and cheat detection.

**Definition 2.7.** *Let $\mathcal{F}$ and $\Pi$ be as above. A protocol $\Pi$ is said to* securely compute *$\mathcal{F}$ with abort and cheat detection if for every non-uniform polynomial-time adversary $\mathcal{A}$ for the real model, there exists a non-uniform polynomial-time adversary $\mathcal{S}$ for the ideal model corrupting the same parties as $\mathcal{A}$, such that*

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

For the simplicity of presentation, we assume that the adversarial behavior is restricted only to premature aborting. Therefore, in the rest of the paper, we replace the "$\texttt{cheat}_j$" message to "$\texttt{abort}_j$" message. We achieve this restriction by using several cryptographic tools such as signatures.

## 2.3 Cryptographic Tools

We next informally describe two cryptographic tools and one standard cryptographic assumption that appear in our protocols. Formal definitions can be found in, e.g., [10, 11].

**Signature Schemes.** A signature on a message proves that the message was created by its presumed sender, and its content was not altered. A signature scheme is a triple $(\mathrm{Gen}, \mathrm{Sign}, \mathrm{Ver})$ containing the key generation algorithm $\mathrm{Gen}$, which gets as an input a security parameter $1^n$ and outputs a pair of keys, the signing key $K_S$ and the verification key $K_v$, the signing algorithm $\mathrm{Sign}$, and the verifying algorithm $\mathrm{Ver}$. We assume that it is infeasible to produce signatures without holding the signing key.

**Secret-Sharing Schemes.** An $\alpha$-out-of-$m$ secret-sharing scheme is a mechanism for sharing data among a set of parties such that every set of parties of size $\alpha$ can reconstruct the secret, while any smaller set knows nothing about the secret. In this paper, we use Shamir's $\alpha$-out-of-$m$ secret-sharing scheme [25]. In this scheme, for every $\alpha - 1$ parties, the shares of these parties are uniformly distributed and independent of the secret. Furthermore, given at most such $\alpha - 1$ shares and a secret $s$, one can *efficiently* complete them to $m$ shares of the secret $s$.

In order to simplify the presentation of our protocols, we construct the following threshold secret-sharing scheme *with respect to a certain party*.

**Construction 2.8.** *Let $s$ be some secret taken from some finite field $\mathbb{F}$. We share $s$ among $m$ parties* with respect to *a special party $p_j$ in an $\alpha$-out-of-$m$ secret-sharing scheme as follows:*

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

**Output:**

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

In the above, we shared the secret $s$ among the parties in $P$ in a secret-sharing scheme such that any set of size at least $\alpha$ that contains $p_j$ can reconstruct the secret. In addition, similarly to the Shamir secret-sharing scheme, the following property holds: for any set of $\beta < \alpha$ parties (regardless if the set contains $p_j$), the shares of these parties are uniformly distributed and independent of the secret. Furthermore, given such $\beta < \alpha$ shares and a secret $s$, one can *efficiently* complete them to $m$ shares of the secret $s$ and *efficiently* select uniformly at random one vector of shares completing the $\beta$ shares to $m$ shares of the secret $s$.

**Trapdoor Permutations.** A trapdoor permutation is a collection of permutations that (1) are easy to compute, (2) hard to invert in polynomial time, and, (3) are easy to invert given some additional key. The existents of trapdoor permutations is a standard cryptographic assumption and it is used in achieving many cryptographic constructions, e.g., public key encryption and oblivious transfer. In Appendix A, we explain that they can be used to construct the multiparty constant-round secure-with-abort protocols that we use in our protocols.

For completeness, we next give an informal definition of trapdoor (one way) permutations. A collection of trapdoor permutations is a set $F = \{f_i : D_i \to D_i\}_{i \in I}$ where $I$ is a set of finite indices and for every $i \in I$, $f_i$ is a permutation such that (1) (Easy to sample function) There exists a probabilistic polynomial time (abbreviated as PPT) algorithm $\mathrm{Gen}$ which on input $1^n$ outputs $(i, t_i)$ where $t_i$ is the trapdoor of $i$, (2) (Easy to sample domain) There exists a PPT algorithm which on input $i \in I$ outputs $x \in D_i$, (3) (Easy to evaluate function) There exists a PPT algorithm $\mathrm{A}$ which on inputs $i \in I$ and $x \in D_i$, computes $\mathrm{A}(i, x) = f_i(x)$, (4) (Hard to invert) Every PPT algorithm fails to invert $y = f_i(x), i$ where $x \in D_i$ with a noticeable probability, and (5) (Easy to invert when trapdoor is known) There exists a PPT algorithm $\mathrm{B}$ such that $\mathrm{B}(i, t_i, f_i(x)) = x$.

## 2.4 The Two-Party Protocol of Moran et al.

Moran, Naor, and Segev [22] present a two-party coin-tossing protocol with an optimal bias with respect to the round complexity (i.e., meeting the lower-bound of Cleve [8]). The MNS protocol uses some of the ideas presented in early works on fairness, e.g., [18, 20, 15, 14]. We next briefly review the MNS protocol, which later serves as the basis for our construction. Following the presentation of the MNS protocol, we first describe a construction that uses an on-line trusted party called the dealer. Later, we describe how the dealer can be eliminated.

The main underlying idea is that the dealer chooses a special round during which the parties actually unknowingly learn the output of the protocol. If the adversary guesses this round, it can bias the output by aborting. If the adversary aborts (or behaves maliciously) in any other time, then there is no bias. However, this special round is uniformly selected (out of $r$ possible rounds) and then concealed such that the adversary is unable to guess it with probability exceeding $1/r$. Therefore, the bias of the protocol is $O(1/r)$.

More specifically, for two parties Alice and Bob to jointly toss a random coin, the protocol proceeds as follows. In a preprocessing phase, the dealer selects a special round number $i^* \in \{1, \ldots, r\}$, uniformly at random, and selects bits $a_1, \ldots, a_{i^*-1}, b_1, \ldots, b_{i^*-1}$, independently and uniformly at random. It then uniformly selects a bit $w \in \{0, 1\}$ and sets $a_i = b_i = w$ for all $i^* \leq i \leq r$. Thereafter, the protocol proceeds in rounds: In round $i$, the dealer gives Alice the bit $a_i$ and Bob the bit $b_i$. If none of the parties abort, then at the end of the protocol both output $a_r = b_r = w$. If a party prematurely aborts in some round $i$, the honest party outputs the bit it received in the previous round (i.e., $a_{i-1}$ or $b_{i-1}$ respectively). If one party aborts before the other party received its first bit (i.e., $a_1$ or $b_1$), then the other party outputs a random bit.

The security of the protocol follows from the fact that, unless the adversary aborts in round $i^*$, it cannot

bias the output of the protocol. The view of any of the parties up to round $i < i^*$ is independent of the value of $i^*$, hence, any adversary corrupting a single party can guess $i^*$ with probability at most $1/r$.

To eliminate the trusted party, Moran et al. first turn the trusted party from an on-line dealer into an off-line dealer, i.e., one that computes some values in a preprocessing phase, gives them to the parties, and halts. To achieve this, they use a 2-out-of-2 secret-sharing scheme and an authentication scheme. The trusted party selects $i^*$, bits $a_1, \ldots, a_{i^*-1}, b_1, \ldots, b_{i^*-1}$, and a bit $w \in \{0,1\}$ as before. It then selects random shares for $a_i$ and $b_i$ for each $i \in \{1, \ldots, r\}$. That is, it selects uniformly at random $a_i^{(A)}, b_i^{(A)} \in \{0,1\}$ and computes $a_i^{(B)} = a_i \oplus a_i^{(A)}$ and $b_i^{(B)} = b_i \oplus b_i^{(A)}$. At the beginning of the protocol, the trusted party sends to Alice her shares of the $a_i$'s, that is, $a_i^{(A)}$, and the shares $b_i^{(A)}$ together with an authentication of the bit $b_i^{(A)}$ with a key $k_A$ (i.e., authenticated shares of the $b_i$'s), and sends to Bob his shares of the $b_i$'s and authenticated shares of the $a_i$'s with a key $k_B$. The key $k_A$ is given to Bob and the key $k_B$ is given to Alice. The protocol now proceeds in rounds. In each round $i$, Bob sends to Alice his authenticated share of $a_i$, so Alice can reconstruct the bit $a_i$. Alice then sends to Bob her authenticated share of $b_i$. An adversary cannot forge with noticeable probability an authentication tag, and is, thus, essentially limited to aborting in deviating from the prescribed protocol.

The off-line dealer is then replaced by the (constant round) secure-with-abort two-party protocol of [21] for computing the preprocessing functionality. That is, at the end of the initialization protocol, the parties obtain the authenticated shares of the $a_i$'s and the $b_i$'s, while the underlying $i^*$ stays secret. The security of the 2-party preprocessing protocol guarantees that a polynomial-time adversary is essentially as powerful as in a computation with an off-line dealer.

# 3 Coin Tossing with a Dishonest Majority – A Warm-Up

In this section we present two warm-up constructions of multiparty coin-tossing protocols with bias $O(1/r)$ where $r$ is the number of rounds in the protocol. The first construction considers the case that at most half of the parties are corrupted (however, there is no majority of honest parties). The second construction solves the problem of coin tossing with 5 parties, where at most 3 are corrupted. These two protocols demonstrate the main difficulties in constructing multiparty coin-tossing protocols with a dishonest majority, alongside the techniques we use to overcome these difficulties. In Sections 4–6, we present a construction for the general case that combines ideas from the two constructions presented in this section.

The main issue of any coin-tossing protocol is how to deal with premature aborts. The protocol must instruct any large enough subset of parties (i.e., at least as large as the set of honest parties) how to jointly reconstruct a bit if all other parties abort the protocol. Since there is no honest majority, an adversary controlling some set of parties can compute the output of this set assuming that the other parties abort. The problem in designing a protocol is how to ensure that this information does not enable the adversary to bias the output.

We stress that constructing fair coin-tossing protocols assuming a trusted dealer is an easy task, e.g., the trusted party can choose a random bit and send it to each party. However, when considering a rushing adversary, one cannot eliminate the trusted party in this protocol. The coin-tossing protocols we describe below are designed such that it is possible to transform them to protocols with no trusted party.

## 3.1 Multiparty Coin Tossing When Half of the Parties are Corrupted

In this section we present a protocol with an optimal (up to a small constant) bias with respect to round complexity, when up to half the parties may be corrupted. We next give an informal description of the

protocol with an on-line trusted party who acts as a dealer. In this protocol, the parties simulate the MNS protocol. That is, we partition the parties into two sets $A$ and $B$, one will simulate Alice and the other will simulate Bob. The main difficulty is that the adversary is not restricted to corrupting parties only in one of these sets. To overcome this problem, in our partition $A$ contains a single party $p_1$, and the set $B$ contains the parties $p_2, \ldots, p_m$. If the adversary corrupts $p_1$, it gains full access to the view of Alice in the 2-party protocol; however, in this case a strict majority of the parties simulating Bob is honest, and the adversary will gain no information about the bits of Bob, i.e., the $b_i$'s.

We next describe the protocol. We assume for simplicity that $m$ is even. In a preprocessing phase, the dealer uniformly selects $i^* \in \{1, \ldots, r\}$ and then uniformly and independently selects $a_1, \ldots, a_{i^*-1}$, $b_1, \ldots, b_{i^*-1}$. Finally, it uniformly selects $w \in \{0, 1\}$ and sets $a_i = b_i = w$ for all $i^* \leq i \leq r$. In each round $i$, the dealer sends $a_i$ to $A$, selects random shares of $b_i$ in Shamir's $m/2$-out-of-$(m-1)$ secret-sharing scheme, and sends each share to the appropriate party in $B$. We stress that formally (to model a rushing adversary), the dealer first sends the corrupted parties their messages, allows them to abort, and proceeds as described below.

During the execution, some parties might abort; we say that a party is active if it has not aborted. If a party $p_j$ prematurely aborts, then the trusted party notifies all currently active parties that $p_j$ has aborted. We next describe the actions when a party aborts:

- If $p_1$ aborts in round $i$, then the parties in $B$ reconstruct $b_{i-1}$, output it, and halt. In this case, since $p_1$ is corrupted, at least $m/2$ honest parties exist in $B$ and, thus, they will be able to reconstruct the output.
- If in round $i$ fewer than $m/2$ active parties remain in $B$, then $p_1$ broadcasts $a_{i-1}$ to the remaining parties in $B$ and all (honest) parties output $a_{i-1}$ and halt. In this case $p_1$ must be honest, and hence, can be trusted to broadcast $a_{i-1}$.
- While there are still at least $m/2$ active parties in $B$ (i.e., at most $m/2 - 1$ of them abort) and $p_1$ is active, the protocol proceeds without a change.

If the last round of the protocol is completed, i.e., $p_1$ got $a_r$ and there are at least $m/2$ active parties in $B$ that got the corresponding shares of $b_r$, then $p_1$ outputs $a_r$ and the active parties in $B$ reconstruct and output $b_r$.

To prevent cheating, the dealer needs to sign the messages given to the parties in $B$ (in our protocols we use signatures and not authentication as in the MNS protocol since if a corrupted party sends an incorrect share we need all honest parties to know this). We omit these details in this subsection.

Recall that at most $m/2$ out of the $m$ parties are corrupted. Thus, if $p_1$ is corrupted, then at most $(m/2) - 1$ parties in $B$ are corrupted, and they cannot reconstruct $b_i$. The argument that the above protocol is $O(1/r)$-secure is now straightforward. An adversary wishing to bias the protocol must cause premature termination. To do so, it must either corrupt $p_1$ (and gain no information on the $b_i$'s) or otherwise corrupt $m/2$ parties in $B$ (hence, leaving $p_1$ uncorrupted). Thus, for any adversary in the multi-party protocol there is an adversary corrupting Alice or Bob in the two party protocol with dealer of the MNS protocol that is, essentially, as powerful. An important feature that we exploit in our protocol is the fact that in the two-party protocol Bob does not need its bit $b_{i-1}$ if Alice does not abort. Thus, in our protocol the parties in $B$ do not reconstruct $b_{i-1}$ unless $p_1$ aborts in round $i$.

More work is required in order to eliminate the trusted dealer; the details are a special case of those described in Section 6. In the above protocol, we partitioned the parties to two sets: $\{p_1\}$ and $\{p_2, \ldots, p_m\}$ and associate a bit to each set. In the general case, we associate bits to subsets; however, the construction is more involved.

**Remark 3.1.** *Our choice to partition the parties into a singleton and a set of size $m-1$ is not the only one possible, and similar ideas would work with any partition of the $m$ parties into two odd-sized disjoint sets. The key observation to be made is that the sets $A$ and $B$ can use $\lceil |A|/2 \rceil$-out-of-$|A|$ and $\lceil |B|/2 \rceil$-out-of-$|B|$ secret sharing of the whole computation of Alice and Bob from the MNS protocol, respectively.*

*If the corrupted parties of either sets reconstruct a secret of the set, then we necessarily have an honest majority in the other set. Furthermore, in the event of premature abort of the majority of parties in one set, the (honest) majority in the other set can complete the execution of the protocol.*

### 3.2 A 5-Party Protocol that Tolerates up to 3 Corrupted Parties

In this section we consider the case where $m = 5$ and $t = 3$, i.e., a 5-party protocol where up to 3 of the parties may be corrupted. The protocol we present in this section is inferior compared to the protocol in Sections 4–6. As in previous protocols, we first sketch our construction assuming there is a special on-line trusted dealer. This dealer interacts with the parties in rounds, sharing bits to subsets of parties, and proceeds with the normal execution as long as at least 4 of the parties are still active.

Denote the trusted dealer by $T$ and the parties by $p_1, \ldots, p_5$. In this protocol, for each interaction round, the dealer produces a bit for each one of the 10 possible sets of three parties; this bit is recovered if a premature termination of at least 2 parties occurs. Formally, for each set $J \subset \{1, \ldots, 5\}$ such that $|J| = 3$, denote $P_J = \{p_j : j \in J\}$ and by $\sigma_J^i$ a bit to be recovered by 2 or 3 active parties in $P_J$ if the protocol terminates in round $i+1$. In a preprocessing phase, the dealer $T$ selects uniformly at random $i^* \in \{1, \ldots, r\}$, indicating the special round in this five-party protocol. Then, for every $0 \leq i < i^*$ it selects $\sigma_J^i$ independently and uniformly at random for each set $J \subset \{1, \ldots, 5\}$ such that $|J| = 3$. Finally, it independently and uniformly selects a random bit $w$ and sets $\sigma_J^i = w$, for every $i \in \{i^*, \ldots, r\}$ and for every set $J \subset \{1, \ldots, 5\}$ such that $|J| = 3$.

The dealer $T$ interacts with $p_1, \ldots, p_5$ in rounds, where round $i$, for $1 \leq i \leq r$ consists of three phases as described below. These phases describe the activity of a rushing adversary in one communication round. Furthermore, for modeling a rushing adversary we assume that the dealer knows the identities of the corrupted parties. In Section 3.2.1 we remove this assumption and the dealer.

**The peeking phase.** The dealer sends to the adversary all the bits $\sigma_J^i$ such that there is a majority of corrupted parties in $P_J$, i.e., at least 2 parties $p_j$ such that $j \in J$ are controlled by the adversary.

**The abort and premature termination phase.** The adversary sends to $T$ a list of parties that abort in the current round. If there are fewer than 4 active parties (i.e., there are either 2 or 3 active parties),[2] $T$ sends $\sigma_{J_L}^{i-1}$ to the active parties, where $J_L$ is the lexicographically first set of three indices that contains all the indices of the active parties. The honest parties output this bit and halt.

**The main phase.** If at least 4 parties are active, $T$ notifies the active parties that the protocol proceeds normally.

If after $r$ rounds, there are at least 4 active parties, $T$ simply sends $w$ to all active parties and the honest parties output this bit.

As an example of a possible execution of the protocol, assume that $p_1$ aborts in round 4 and $p_3$ and $p_4$ abort in round 37. In this case, $T$ sends $\sigma_{\{1,2,5\}}^{36}$ to $p_2$ and $p_5$, which output this bit.

[2]The reason for requiring that the dealer does not continue when at least two parties abort will become clear when we transform the protocol to a protocol with an off-line dealer.

Recall that the adversary obtains the bit of the set of size three of parties indexed by $J_L$ if at least two parties in $P_J$ are corrupted. If the adversary causes the dealer to halt, then, either there are two remaining active parties, both of them must be honest, or there are three active parties and at most one of them is corrupted. In either case, the adversary does not know $\sigma_J^{i-1}$ in advance. Furthermore, the dealer reveals the appropriate bit $\sigma_J^{i-1}$ to the active parties. Jumping ahead, these properties are later preserved in a real world protocol by using a 2-out-of-3 secret-sharing scheme.

We next argue that any adversary can bias the output of the above protocol by at most $O(1/r)$. As in the MNS protocol, the adversary can only bias the output by causing the protocol to terminate in round $i^*$. In contrast to the MNS protocol, in our protocol if in some round there are two bits $\sigma_J^i$ and $\sigma_{J'}^i$ that the adversary can obtain such that $\sigma_J^i \neq \sigma_{J'}^i$, then the adversary can deduce that $i \neq i^*$. However, there are at most 7 bits that the adversary can obtain in each round (i.e., the bits of sets $P_J$ containing at least two corrupted parties). For a round $i$ such that $i < i^*$, the probability that all these bits are equal to (say) 0 is $2^{-7}$. Such rounds are indistinguishable to the adversary from round $i^*$. Intuitively, the best an adversary can do is guess one of these rounds, and therefore cannot succeed guessing with probability better than $2^{-7}$. Thus, the bias the adversary can cause is $2^7/r$. This intuition is proved in Claim 5.5 (Lemma 2 from [17]).

### 3.2.1 Eliminating the Dealer for the 5-Party Protocol

We eliminate the trusted on-line dealer in a few steps using a few layers of secret-sharing schemes. First, we change the on-line dealer, so that in each round $i$ it shares the bit $\sigma_J^i$ among the parties in $P_J$ using a 2-out-of-3 secret-sharing scheme – called *inner* secret-sharing scheme. In fact, the dealer first signs the bit share to prevent cheating. The same requirements on $\sigma_J^i$ as in the above protocol are preserved using this inner secret-sharing scheme. That is, the adversary is able to obtain information on $\sigma_J^i$ only if it controls at least two of the parties in $P_J$. On the other hand, if the adversary does not control at least two parties in some $P_J$ (i.e., there is an honest majority in $P_J$), then, in round $i$, the honest parties can reconstruct $\sigma_J^{i-1}$ (if so instructed by the protocol).

Next we turn the on-line dealer into an off-line dealer. That is, we show that it is possible for the dealer to only interact with the parties once, sending each party some input, so that thereafter, the parties interact in rounds (without the dealer) and in each round $i$ each party learns its shares in the $i$th inner secret-sharing scheme. That is, in each round $i$, each party $p_j$ learns a share of $\sigma_J^i$ in a 2-out-of-3 secret-sharing scheme, for every set $J$ such that $|J| = 3$ and $j \in J$. For this purpose, the dealer shares, in a preprocessing phase, the appropriate shares for the inner secret-sharing scheme using a 4-out-of-5 secret-sharing scheme with respect to $p_j$ as defined in Construction 2.8. We call the resulting secret-sharing scheme the *outer* scheme.

The use of the 4-out-of-5 secret-sharing scheme plays a crucial role in eliminating the on-line dealer. It guarantees that an adversary, corrupting at most three parties, cannot reconstruct the shares of round $i$ before round $i$. However, if at least two parties do not reveal their shares, then they prevent a reconstruction of the outer scheme (this is why we cannot proceed after 2 parties aborted); in this case there is an honest majority among the active parties . Hence, the protocol proceeds normally as long as at least 4 parties are active. If, indeed, at least two parties abort (in round $i$), then the remaining parties use their shares of the inner scheme to reconstruct the bit $\sigma_{J_L}^{i-1}$ for the appropriate set $J_L$ of size three.

To prevent corrupted parties from cheating, by say, sending false shares and causing reconstruction of wrong secrets, every message that a party should send during the execution of the protocol is signed in the preprocessing phase (together with the appropriate round number and with the party's index). Furthermore, all shares in the inner secret-sharing scheme are signed (as they are used as messages if reconstruction is required). In addition, the dealer sends a verification key to each of the parties. To conclude, the off-line dealer gives each party the signed shares for the outer secret-sharing scheme together with the verification

key. A formal description of the functionality of the off-line dealer, called $\mathrm{ShareGenForFive}_r$, is given in Figure 1. In order to simplify the presentation, we use slightly different notations compared to the notations used in the presentation of the general construction in Sections 4–6.

**Functionality** $\mathrm{ShareGenForFive}_r$

**Notations:** $1^n$ – the security parameter and $r = r(n)$ – the number of rounds in the protocol.

**Computing default bits and signing keys**

1. Choose $w \in \{0,1\}$ and $i^* \in \{1,\ldots,r\}$ uniformly at random.
2. For each $i \in \{1,\ldots,r\}$ and for each set $J \subset \{1,\ldots,5\}$ such that $|J| = 3$,
   (a) if $i \leq i^* - 1$, then choose independently and uniformly at random $\sigma_J^i \in \{0,1\}$.
   (b) if $i \geq i^*$, then set $\sigma_J^i = w$.
3. Compute $(K_{\mathrm{sign}}, K_{\mathrm{ver}}) \leftarrow \mathrm{Gen}(1^n)$.

**Computing signed shares of the inner secret-sharing scheme**

4. For each $i \in \{1,\ldots,r\}$ and for each set $J \subset \{1,\ldots,5\}$ such that $|J| = 3$,
   (a) Share $\sigma_J^i$ in an 2-out-of-3 Shamir's secret-sharing scheme for the parties in $P_J$. For each party $p_j \in P_J$, let $S_j^{i,J}$ be its share of $\sigma_J^i$.
   (b) Sign each share $S_j^{i,J}$: $R_j^{i,J} \leftarrow (S_j^{i,J}, i, J, j, \mathrm{Sign}((S_j^{i,J}, i, J, j), K_{\mathrm{sign}}))$.

**Computing shares of the outer secret-sharing scheme**

5. For each $i \in \{1,\ldots,r\}$, for each set $J \subset \{1,\ldots,5\}$ such that $|J| = 3$, and for each $j \in J$: share $p_j$'s signed share $R_j^{i,J}$ using a 4-out-of-5 secret-sharing scheme with respect to $p_j$ as defined in Construction 2.8; that is, one masking share $\mathrm{mask}_j(R_j^{i,J})$ that only $p_j$ obtains and 4 complement shares for the rest of the parties are produced.

**Signing the messages of all parties**

6. For every $1 \leq q \leq 5$, compute the message $m_{(q,i)}$ that $p_q \in \{p_1,\ldots,p_5\}$ broadcasts in round $i$ by concatenating (1) $q$, (2) the round number $i$, and (3) the complement shares $\mathrm{comp}_q(R_j^{i,J})$ for all sets $J$ such that $|J| = 3$ and for all $j \neq q$ such that $j \in J$ produced in Step (5) for $p_q$.
7. Compute $M_{q,i} \leftarrow (m_{q,i}, \mathrm{Sign}(m_{q,i}, K_{\mathrm{sign}}))$.

**Outputs:** Each party $p_j$ receives

- The verification key $K_{\mathrm{ver}}$.
- The messages $M_{j,1},\ldots,M_{j,r}$ that $p_j$ broadcasts during the protocol.
- $p_j$'s private masks $\mathrm{mask}_j(R_j^{i,J})$ which were produced in Step (5) for each $1 \leq i \leq r$ and each set $J \subset \{1,\ldots,5\}$ such that $|J| = 3$ and $j \in J$.

Figure 1: The initialization functionality $\mathrm{ShareGenForFive}_r$.

The protocol with the off-line dealer proceeds in rounds. In round $i$ of the protocol, all parties broadcast their (signed) shares in the outer (4-out-of-5) secret-sharing scheme. Thereafter, each party can unmask the message it receives (with its share in the appropriate 2-out-of-2 secret-sharing scheme) to obtain its signed

shares in the 2-out-of-3 sharing of the bits $\sigma_J^i$ (for the appropriate sets $J$'s such that $|J| = 3$ for which the party belongs). If a party stops broadcasting messages or broadcasts improperly signed messages, then all other parties consider it as aborted and will ignore all messages it will send in the future. If two or more parties abort, the remaining parties reconstruct the bit of the lexicographically first set of size three that contains all of them, as described above. In the special case of premature termination already in the first round, the remaining parties engage in a fully-secure protocol (with honest majority) to toss a completely random coin. In Figure 2 we formally define the 5-party coin-tossing protocol tolerating up-to 3 corrupted parties.

**A 5-party protocol** $\mathrm{FivePartyCoinToss}_r$

**Notations:** $1^n$ – the security parameter and $r = r(n)$ – the number of rounds in the protocol.

**Joint input:** Security parameter $1^n$.

**Preliminary phase:**

- The parties engage in a secure-with-abort and cheat-detection protocol in order to computing the functionality $\mathrm{ShareGenForFive}_r$. If a single party aborts, then this phase is repeated. This may happen at most twice before an honest majority is guaranteed. If at least two parties aborted (and detected), the parties use multiparty secure computation (with full security) to toss a fair coin, output this resulting bit, and halt.

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

1. Each active party $p_j \in \{p_1, \ldots, p_5\}$ broadcasts $M_{(j,i)}$ (its share of the $p_q$'s share of the bit $\sigma_J^i$ for each set $J \subset \{1, \ldots, 5\}$ such that $|J| = 3$ and each $q \in J$ where $q \neq j$)
2. For each active party $p_j \in \{p_1, \ldots, p_5\}$, if $\mathrm{Ver}(M_{(j,i)}, K_{\mathrm{ver}}) = \mathrm{Fail}$ or if $p_j$ broadcasts an invalid or no message, then all parties mark $p_j$ as inactive.
3. If at least 2 parties were marked inactive:
   - If $i = 1$ (i.e., the parties aborted during the first round), then the active parties use a multiparty secure protocol (with full security) to toss a fair coin, output this resulting bit, and halt.
   - Otherwise, the active parties reconstruct the bit $\sigma_{J_L}^{i-1}$ where $J_L$ is the lexicographically first set of size three that contains all the indices of the active parties, output it, and halt.

**At the end of round** $r$**:** Each active party $p_j$ broadcasts the signed share $R_j^{r,J}$, for each set $J \subset \{1, \ldots, 5\}$ such that $|J| = 3$ and $j \in J$, in order to reconstruct $\sigma_J^r = w$. All honest parties output $w$.

Figure 2: The $\mathrm{FivePartyCoinToss}_r$ protocol.

Finally, we replace the off-line dealer by using a secure-with-abort protocol with cheat detection (see Section 2.2) computing the functionality computed by the dealer, that is, Functionality $\mathrm{ShareGenForFive}_r$. Obtaining the outputs of this computation, an adversary is unable to infer any information regarding the output of the coin-tossing protocol (as the functionality does not give any information to an adversary controlling at most 3 parties). The adversary, however, can prevent the execution, at the price of at least one corrupted party being detected by all other parties. In such an event, the remaining parties will start over.

This can go on at most twice, before there is a honest majority and a completely fair coin can be obtained. The details of this final step are given in Section 6.

**Remark 3.2.** *The above construction can be generalized in a straightforward manner to any number $m$ of parties and any number $t$ of corrupted parties such that $t < 2m/3$. In the generalized construction, there is a bit for every set $Q$ of size $t$; the outer secret-sharing scheme is a $(t+1)$-out-of-$m$ scheme and the inner secret-sharing scheme is an $(m-t)$-out-of-$t$ scheme. The bias of the resulting protocol is much higher compared to the protocol described in Sections 4–6, which uses a better way for distributing bits to subsets. The bias of the generalized protocol and the protocol presented in Sections 4–6 is $O(2^{\alpha}/r)$, where $\alpha$ is the number of bits the adversary sees. In the protocol presented in Sections 4–6, $\alpha = 2^{k+1}$. We can show that in the generalized protocol $\alpha$ is much higher, that is, $\alpha > (m-t)2^k$. Therefore, the bias of the protocol is much higher than $2^{(m-t)2^k}/r$.*

# 4 Our Main Construction

In Sections 4–6 we present our main result – a coin-tossing protocol that has nearly optimal bias and can tolerate up to $2/3$ fraction of corrupted parties. More specifically, we prove the following theorem.

**Theorem 4.** *Let $n$ be the security parameter. If enhanced trapdoor permutations exist, then for any $m = m(n)$, $t = t(n)$, and $r = r(n)$ such that $m/2 \leq t < 2m/3$, there is an $r$-round $m$-party coin-tossing protocol tolerating up to $t$ corrupted parties that has bias $O\left(2^{2^{k+1}}/r'\right)$, where $k = 2t - m$ and $r' = r - O(k+1)$.*

In the above theorem, $k$ is the difference between the number of corrupted parties and the number of honest parties, i.e., $k = t - (m-t) = 2t - m$. For example, if $t = m/2$, then the number of honest and corrupted parties are equal and $k = 0$. Furthermore, for concreteness the theorem assumes that trapdoor permutations exist. We can replace this assumption with ans assumption that implies a constant round secure-with-abort with cheat detection exists.

Following the MNS protocol, we describe our protocol in two steps. In Section 5, we describe Protocol $\mathrm{CTWithD}_r$ that uses an online trusted party, called dealer. In Section 6, we eliminate the on-line dealer. This simplifies the description and understanding of our protocols. More importantly, we prove the security of our main protocol in a modular way. In Section 5, we prove the following theorem.

**Theorem 5.** *Protocol $\mathrm{CTWithD}_r$ is an $r$-round $m$-party coin-tossing protocol with an on-line dealer tolerating up to $t$ corrupted parties and has bias $O\left(2^{2^{k+1}}/r\right)$.*

We then consider the on-line dealer of Protocol $\mathrm{CTWithD}_r$ as an ideal functionality. In this protocol, the honest parties do not send any messages and in each round the dealer sends messages to the parties; we consider an interactive functionality sending the messages that the dealer sends. In Section 6, we prove the following theorem.

**Theorem 6.** *Let $n$ be the security parameter. Let $m = m(n)$, $t = t(n)$, and $r = r(n)$ such that $t < 2m/3$ and $k = 2t - m$. If enhanced trapdoor permutations exist, then Protocol $\mathrm{CoinToss}_{r'}$ presented in Section 6, is a computationally-secure implementation (with full security) of the dealer's functionality in Protocol $\mathrm{CTWithD}_r$. Protocol $\mathrm{CoinToss}_{r'}$ has $r' = r + O(k+1)$ rounds.*

*Proof of Theorem 4.* This theorem follows from Theorem 5 and Theorem 6 by the composition theorem of Canetti [6]. Formally, consider the (ideal) functionality $\mathcal{F}_D$ of the dealer in Protocol $\mathrm{CTWithD}_r$. By Theorem 6, Protocol $\mathrm{CTWithD}_r$ is a computationally-secure implementation of $\mathcal{F}_D$ (with full security). Consider the protocol $\Pi^{\mathcal{F}_D}$ which simply executes the functionality $\mathcal{F}_D$. By Theorem 5, $\Pi^{\mathcal{F}_D}$ is a $1/p$-secure implementation of $\mathrm{CoinToss}()$. By the composition theorem of Canetti [6], Protocol $\Pi^{\mathrm{CTWithD}_r}$ is a $1/p$-secure implementation of $\mathrm{CoinToss}()$. □

# 5 A Protocol with an On-Line Dealer

In this section we describe a protocol with a special trusted party $T$ that acts as an on-line dealer interacting with the parties in rounds. The protocol consists of $r$ rounds of interaction. In a preliminary phase, the trusted party $T$ selects the prescribed output bit $w$ and a special round number $i^*$ uniformly at random. In each round of interaction, the trusted party $T$ selects a bit for each subset in a predetermined collection (this collection is part of the design of the protocol). The bits of all subsets for any round before round $i^*$ are uniformly and independently chosen; the bits of all subsets for all rounds from round $i^*$ and on all equal $w$. These bits are not given to honest parties. However, if the number of corrupted parties in a subset $Q$ is larger than some predefined threshold, then the corrupted parties get the bit of $Q$.

In the real world these bits are secret shared. Since in the real world the adversary is rushing, the corrupted parties in the real world receive their shares before the honest parties do. Hence, in the protocol with the dealer, interaction between the parties and $T$ in each round has three phases. In the first phase, for each subset $Q$ that contains enough corrupted parties, the trusted party sends the bit of the subset to the corrupted parties in the subset $Q$. In the second phase, corrupted parties may abort the computation (and by that prevent later reconstruction of some of the information). To do so, these parties send to $T$ an "abort" message. Finally, in the third phase, a "proceed" message is sent to the parties (this can be considered as an "ideal" secret-sharing).

The protocol proceeds normally as long as fewer than $m - t$ parties abort. If at most $t$ parties are active in round $i$, the trusted party reveals the value of the bit of some subset indexed by $J$ in round $i - 1$, i.e., $\sigma_J^{i-1}$. The subset subset indexed by $J$ is chosen in a way that guarantees that the adversary does not know the value of this bit before instructing parties to abort. Jumping ahead, in the real world the subset indexed by $J$ will also hold the additional property that the adversary cannot prevent the reconstruction of the bit of $Q$ from round $i - 1$. The exact underlying subsets are specified in Figure 3. As an example, consider the case where $m = 5$ and $t = 3$. In this case, there are 3 underlying subsets: $P_1 = \{p_1\}$, $P_2 = \{p_2\}$, and $P_3 = \{p_3, p_4, p_5\}$ and three additional sets: $Q_{\{1,2\}} = \{p_1, p_2\}$, $Q_{\{1,3\}} = \{p_1, p_3, p_4, p_5\}$, and $Q_{\{2,3\}} = \{p_2, p_3, p_4, p_5\}$. Observe that the protocol appears in Section 3.2 is also designed for the case of $m = 5$ and $t = 3$, but in that protocol, there are 10 sets for which the trusted party sends a value in each round. Roughly speaking, keeping the number of sets small is an important target, as there is a connection between this number and the bias of the protocol, as we explain below.

In order to simplify the presentation of this protocol, we assume that there are private channels between each party and the on-line dealer. The formal description of the protocol called $\mathrm{CTWithD}_r$ is given in Figure 3. In Sections 5.1–5.2, we prove that Protocol $\mathrm{CTWithD}_r$, is an $O(2^{2^{k+1}}/r)$-secure protocol for computing the $\mathrm{CoinToss}()$ functionality. Our proof follows the real vs. ideal paradigm. We consider a malicious adversary $\mathcal{A}$ that corrupts at most $t < 2m/3$ of the parties in an execution of Protocol $\mathrm{CTWithD}_r$. In Section 5.1 we present a simulator $\mathcal{S}$ that corrupts the same subset of parties in the ideal world with access to the trusted party $T_{\mathrm{CoinToss}}$ for computing the $\mathrm{CoinToss}()$ functionality (that is, $T_{\mathrm{CoinToss}}$ gives the same uniformly chosen bit to all parties). The simulator $\mathcal{S}$ has black-box access to $\mathcal{A}$ and at the end of the

computation it outputs a view of the adversary $\mathcal{A}$. In Section 5.2 we bound the statistical distance between the random variable describing the view of the real-world adversary $\mathcal{A}$ concatenated with the output of the honest parties in the appropriate execution and the random variable describing the output of the simulator $\mathcal{S}$ concatenated with the bit selected by the ideal-world trusted party.

**The bias of the protocol – an informal argument.** We next informally explain why the protocol has small bias, that is, we give a sketch of the proof of Theorem 5. First, we claim that the adversary can bias the output only if the premature termination occurs in round $i^*$:

1. If the premature termination occurs after round $i^*$ (or does not occur at all), then the output is already fixed.

2. If the premature termination occurs before round $i^*$, then the adversary does not know the random bit $\sigma_J^{i-1}$, which is the output of the honest parties, as there are fewer than $o_J$ corrupted parties in $Q_J$ (as shown in the formal analysis of this event in Section 5.2).

Thus, the adversary can bias the output only if it guesses $i^*$. If $\sigma_J^i \neq \sigma_{J'}^i$ for two bits that the adversary gets from the trusted party, then it can learn that $i < i^*$. In Claim 5.4 we show that the adversary gets $2^{k+1}$ such bits out of the $2^{k+2}$ bits (remember that there are $k+2$ subsets and bits are give to each union of these subsets). With probability $1/2^{2^{k+1}}$, all these bits are all equal in a round prior to $i^*$ and the adversary cannot distinguish such round from $i^*$. By Claim 5.5 (originally proved in [17, Lemma 2]), this implies that the adversary can guess $i^*$ with probability at most $2^{2^{k+1}}/r$. Therefore, the bias is $O(2^{2^{k+1}}/r)$.

**Remark 5.1.** *In our protocol, we construct a collection of sets $Q_J$ for $J \subset \{1, \ldots, k+2\}$ and give a bit to every set $Q_J$. Our choice of constructing these sets is not the only one possible. For example, as explained in Remark 3.1, we can use the collection of all sets of size $t$ (this leads to a protocol with a larger bias).*

*The main property that is essential for achieving a coin-tossing protocol in our framework is that the collection of the sets and the corresponding thresholds satisfy that every set of at most $t$ active parties that contains all honest parties can choose a set of parties $Q$ from the collection, which has threshold $o$, such that there are at least $o$ active parties in the set $Q$, while the number of corrupted parties in that set is less than $o$. Thus, in each round of the protocol, the honest parties can reconstruct a value, that the corrupted parties cannot see in advance.*

*In Protocol $\mathrm{CTWithD}_r$, we construct the collection of sets by partitioning the $m$ parties into sets $P_1, \ldots, P_{k+2}$ and considering all the unions of these $k+2$ sets. We can show that every partition of the $m$ parties into $\alpha$ odd-sized disjoint sets, where $\alpha \geq k+2$, leads to a valid protocol. However, we can show that constructing the collection by first partitioning the parties into disjoint sets and then taking all unions (provided that the above condition holds) yields a protocol in which the adversary can see at least $2^{k+1}$ bits. That is, in such construction, the bias of the resulting protocol is at least the bias of Protocol $\mathrm{CTWithD}_r$.*

*To conclude, the partition approach for constructing the collection cannot improve the bias of the protocol. We do not know if there is a different approach for constructing the collection that can reduce the bias (i.e., improve the fraction of $2^{2^{k+1}}$ in the bias).*

## 5.1 The Simulator for the On-Line Dealer Protocol

We next describe our simulator $\mathcal{S}$ for Protocol $\mathrm{CTWithD}_r$. Let $B$ be the subset of corrupted parties in the execution and $\mathcal{A}$ a real-world adversary corrupting them. On input $(1^n, \mathrm{aux})$, the simulator $\mathcal{S}$ obtains $w_S = \mathrm{CoinToss}()$ from the trusted party $T_{\mathrm{CoinToss}}$ (i.e., $w_S$ is uniformly selected from $\{0,1\}$). The simulator

**Notations:** $1^n$ – the security parameter, $r = r(n)$ – the number of rounds in the protocol, $m = m(n)$ – the number of parties, $t = t(n)$ – a bound on the number of corrupted parties, and $k$ – the difference between the number of corrupted parties and the number of honest parties.

**Joint Input:** The security parameter $1^n$.

**Underlying Subsets:** Let $P_j = \{p_j\}$ for $1 \leq j \leq k+1$ and $P_{k+2} = \{p_{k+2}, \ldots, p_m\}$.
Define $Q_J = \cup_{j \in J} P_j$ for each $J \subset \{1, \ldots, k+2\}$.

For each subset $P_j$ define a reconstruction threshold value $o_j$: For $1 \leq j \leq k+1$ define $o_j = 1$ and define $o_{k+2} = m - t$. Finally, define $o_J = \sum_{j \in J} o_j$ for each $J \subset \{1, \ldots, k+2\}$.

**Instructions for the (trusted) dealer:**

**The preprocessing phase:** Select a bit $\sigma_J^i$ for each subset $Q_J$ and each round $i$ as follows:

1. Select $i^* \in \{1, \ldots, r\}$ and $w \in \{0, 1\}$ independently with uniform distribution.
2. For each $J \subset \{1, \ldots, k+2\}$, select $\sigma_J^0, \ldots, \sigma_J^{i^*-1}$ independently with uniform distribution.
3. For each $J \subset \{1, \ldots, k+2\}$, set $\sigma_J^{i^*} = \cdots = \sigma_J^r = w$.

**Protocol initialization:** Send a "start" message to all parties. For each party $p_j$, upon receiving an "$\texttt{abort}_j$" from party $p_j$, notify all parties that party $p_j$ has aborted. If at least $m - t$ parties aborted, go to premature termination with $i = 1$. Otherwise, send "proceed" to all parties.

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

- **The peeking phase:** For each $J \subset \{1, \ldots, k+2\}$, if $Q_J$ contains at least $o_J$ corrupted parties, send the bit $\sigma_J^i$ to all corrupted parties in $Q_J$.
- **The abort phase:** For each party $p_j$, upon receiving an "$\texttt{abort}_j$" message from party $p_j$, notify all parties that party $p_j$ has aborted. (Ignore all other types of messages.)
  If at least $m - t$ parties have aborted so far, go to premature termination step.
- **The main phase:** Send "proceed" to all parties.

**Premature termination step:** This step consists of two phases, after which the protocol terminates and all honest parties hold the same output.

- **The abort phase:** For each party $p_j$, upon receiving an "$\texttt{abort}_j$" message from party $p_j$, remove party $p_j$ from the list of active parties.
- **The default output phase:** Let $D$ be the set of indices of parties that aborted the protocol thus far, i.e., $D = \{j \ : \ p_j \text{ has aborted}\}$.
  - If $|D \cap \{k+2, \ldots, m\}| \geq m - t$ then $J = \{1, \ldots, k+1\} \setminus D$.
  - If $|D \cap \{k+2, \ldots, m\}| < m - t$ then $J = (\{1, \ldots, k+1\} \setminus D) \cup \{k+2\}$.
  - Send $w' = \sigma_J^{i-1}$ to all parties and halt.

**Normal termination:** This phase is executed if the last round of the protocol is completed. Send $w$ to all parties.

**Instructions for honest parties:** Upon receiving output $y$ from the dealer, output $y$. (Honest parties do not send any message throughout the protocol.)

Figure 3: Protocol $\mathrm{CTWithD}_r$.

$\mathcal{S}$ internally invokes $\mathcal{A}$ with input $(1^n, \text{aux})$. $\mathcal{S}$ interacts with $\mathcal{A}$ playing the role of $T$, by proceeding as follows:

**Simulating the preprocessing phase:** $\mathcal{S}$ selects a bit $\sigma_J^i$ for every subset $Q_J$ for which the corrupted parties get $\sigma_J^i$ (i.e., for which $|B \cap Q_J| \geq o_J$), and every round $i$ as follows:

1. Selects $i^* \in \{1, \ldots, r\}$ uniformly at random.
2. For each $J \subset \{1, \ldots, k+2\}$, the simulator $\mathcal{S}$ selects uniformly and independently at random $\sigma_J^1, \ldots, \sigma_J^{i^*-1} \in \{0,1\}$.
3. For each $J \subset \{1, \ldots, k+2\}$, the simulator $\mathcal{S}$ sets $\sigma_J^{i^*} = \ldots = \sigma_J^r = w_S$.

**Simulating protocol initialization:** $\mathcal{S}$ sends a "start" message to all corrupted parties (that is, $\mathcal{S}$ sends the messages for the corrupted parties to the adversary $\mathcal{A}$). For each party $p_j$, upon receiving an "$\texttt{abort}_j$" message from party $p_j$, the simulator notifies all corrupted parties that party $p_j$ has aborted. If at least $m - t$ parties aborted, $\mathcal{S}$ sets $i = 1$ and proceeds to simulate the premature termination step. Otherwise, $\mathcal{S}$ sends "proceed" to all corrupted parties.

**Simulating interaction rounds:** In each round $1 \leq i \leq r$, the simulator $\mathcal{S}$ interacts with the corrupted parties, in three phases:

- **The peeking phase:** For each $J \subset \{1, \ldots, k+2\}$, if $Q_J$ contains at least $o_J$ corrupted parties, then $\mathcal{S}$ sends the bit $\sigma_J^i$ to all corrupted parties in $Q_J$ (i.e., to the adversary).
- **The abort phase:** For each party $p_j$, upon receiving an "$\texttt{abort}_j$" message from party $p_j$, the simulator notifies all parties that party $p_j$ is inactive.

  If at least $m - t$ parties have aborted so far, $\mathcal{S}$ simulates the premature termination step.
- **The main phase:** $\mathcal{S}$ sends "proceed" to all parties.

**Simulating the premature termination step:** This round consists of two phases, after which the simulation terminates.

- **The abort phase:** For each party $p_j$, upon receiving an "$\texttt{abort}_j$" message from party $p_j$, remove party $p_j$ from the list of active parties.
- **The default output phase:** Regardless of the subset of set of aborted parties, the simulator sends $w_S$ to all parties and halts the interaction with $\mathcal{A}$.

**Simulating normal termination:** If the last round of the protocol is completed, then $\mathcal{S}$ sends $w_S$ to all parties.

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

## 5.2 Proof of the Correctness of the Simulation

We consider the two random variables from Section 2.1, both of the form $(V, C)$, where $V$ describes a possible view of $\mathcal{A}$, and $C$ describes a possible output of the honest parties (i.e., $C \in \{0,1\}$). The first random variable $\text{REAL}_{\text{CTWithD}_r, \mathcal{A}(\text{aux})}(1^n)$ describes the real world – an execution of the $\text{CTWithD}_r$ protocol, where $V$ describes the view of the adversary $\mathcal{A}$ in this execution, and $C$ is the output of the honest

parties in this execution. The second random variable $\mathrm{IDEAL}_{\mathrm{CoinToss}(),\mathcal{S}(\mathrm{aux})}(1^n)$ describes the ideal world – an execution with the trusted party $T_{\mathrm{CoinToss}}$, where $V$ describes the output of the simulator $\mathcal{S}$ in this execution, and $C$ is the output of the honest parties in this execution. Our goal here is to show that the statistical distance between these two random variables is $O(2^{2^{k+1}}/r)$, that is, to show that the protocol is *perfectly* $O(2^{2^{k+1}}/r)$ secure. For the rest of this proof, we simplify notations by omitting notations that are clear from the context. In particular, we denote the above two random variables by $\mathrm{REAL} = (V_{\mathrm{REAL}}, C_{\mathrm{REAL}})$ and $\mathrm{IDEAL} = (V_{\mathrm{IDEAL}}, C_{\mathrm{IDEAL}})$ respectively.

**Lemma 5.2.** *For every non-uniform polynomial-time adversary $\mathcal{A}$ corrupting at most $t < 2m/3$ of the parties in Protocol $\mathrm{CTWithD}_r$, for the simulator $\mathcal{S}$ described in Section 5.1 (controlling the same parties as $\mathcal{A}$), for every $n \in \mathbb{N}$ and for every $\mathrm{aux} \in \{0,1\}^*$, the following holds:*

$$\mathrm{SD}\left(\mathrm{IDEAL}_{\mathrm{CoinToss}(),\mathcal{S}(\mathrm{aux})}(1^n), \mathrm{REAL}_{\mathrm{CTWithD}_r,\mathcal{A}(\mathrm{aux})}(1^n)\right) = O(2^{2^{k+1}}/r).$$

*Proof.* The flow of our proof is as follows. We first show that statistical distance between the two random variables is at most the probability that the adversary $\mathcal{A}$ guesses the special round $i^*$. We do this by showing that, conditioned on the event that the adversary fails to guess round $i^*$, the two random variables are identically distributed. Then, we bound the probability of guessing $i^*$. We show that the adversary learns at most $2^{k+1}$ new bits in each round. Hence, the probability that all these bits are equal in round $i < i^*$ is at least $\frac{1}{2^{2^{k+1}}}$. Finally, we show that this implies that the probability that of guessing $i^*$ correctly is bounded by $2^{2^{k+1}}/r$.

We consider the probability of a given pair $(v, c)$ according to the two different random variables. For this we recall that the honest parties are deterministic in both models. We compare the two following probabilities:

1. The probability that $v$ is the view of the adversary $\mathcal{A}$ in an execution of Protocol $\mathrm{CTWithD}_r$ and $c$ is the output of the honest parties in this execution, i.e.,

   $$\Pr[\mathrm{REAL} = (v, c)] = \Pr[V_{\mathrm{REAL}} = v \quad \wedge \quad C_{\mathrm{REAL}} = c],$$

   where the probability is taken over the random coins of the dealer $T$.

2. The probability that $v$ is the output of the simulator $\mathcal{S}$ in an ideal-world execution with the trusted party $T_{\mathrm{CoinToss}}$ and $c$ is the output of the honest parties in this execution, i.e.,

   $$\Pr[\mathrm{IDEAL} = (v, c)] = \Pr[V_{\mathrm{IDEAL}} = v \quad \wedge \quad C_{\mathrm{IDEAL}} = c],$$

   where the probability is taken over the random coins of the simulator $\mathcal{S}$ and the random coins of the ideal-world trusted party $T_{\mathrm{CoinToss}}$.

Observe that in the simulation $\mathcal{S}$ follows the same instructions as the trusted party $T$ in Protocol $\mathrm{CTWithD}_r$, except for two changes. First, $\mathcal{S}$ does not flip an internal coin to obtain $w_S$ (in contrast $T$ chooses $w$ at random), but rather gets $w_S$ externally from $T_{\mathrm{CoinToss}}$. However, since both $w_S$ and $w$ are uniformly distributed, it is the same operation. The major difference is that in the case of a premature termination step, $\mathcal{S}$ always uses $w_S$ as its message to the corrupted parties, while $T$ uses the bit of the appropriate subset $Q_J$ as its message. In both cases, this last message sent to the adversary is simply the output bit of the honest parties, that is, in both of the above random variables, the last message in $v$ equals $c$. Hence, it is equivalent to consider the two random variables as above, where in both cases, the last message sent to

the corrupted parties in the premature termination round is omitted from $c$. Thus, we consider the possible views $v$ of the adversary without the last message sent to the corrupted parties.

We define a random variable $\mathrm{PT} \in \{\text{“before”}, \text{“during”}, \text{“after”}\}$ specifying whether premature termination occurred before, during, or after the special round $i^*$. I.e., let $i$ be the round number in which premature termination occurs (if ever); if $i < i^*$ (or if premature termination occurs during the protocol initialization step), then $\mathrm{PT} =$ “before”, if $i = i^*$, then $\mathrm{PT} =$ “during”, and if $i > i^*$ or premature termination never occurs, then $\mathrm{PT} =$ “after” . The random variable is defined both in the ideal and the real world.

In the next claim, we bound the statistical distance between the two random variables according to Definition 2.1.

**Claim 5.3.** $\mathrm{SD}(\mathrm{IDEAL}, \mathrm{REAL}) \leq \frac{1}{2} \Pr[\mathrm{PT} = \text{“during”}]$.

In order to prove this claim, we analyze the above probabilities conditioned on these three possible events:

**$\mathrm{PT} =$ “before” – premature termination occurs before round $i^*$.** We argue that in this case, both in the real protocol and in the simulation, the actions of $\mathcal{A}$ (according to $V$) and its view before the premature termination step are independent of $C$ (and, hence, also of the message sent to corrupted parties in the premature termination) and thus, the probabilities are exactly the same.

In a real-world execution, the output of the honest parties is the random bit for the appropriate set $Q_J$, i.e., $C_{\mathrm{REAL}} = \sigma_J^{i-1}$. This bit is independent of all other bits (since $i - 1 < i^*$). Hence, if $\mathcal{A}$ does not receive $\sigma_J^{i-1}$ from $T$, then this bit is completely unknown to $\mathcal{A}$ (before the last message is sent to the corrupted parties). This is clearly true if premature termination occurs during the protocol initialization. To see that is is also the case if premature termination occurs during round $i < i^*$, consider the following two cases:

1. If $|D \cap \{k+2, \ldots, m\}| \geq m - t$, then $J = \{1, \ldots, k+1\} \setminus D$ and $o_J = |J|$. There are at most $t$ corrupted parties and at least $m - t$ of them are in $Q_{\{k+2\}}$, thus, at most $t - (m - t) = k$ corrupted parties among the $k+1$ parties $p_1, \ldots, p_{k+1}$. In other words, there is at least one honest (and therefore active) party in $Q_J$, and the adversary does not learn $\sigma_J^{i-1}$ during the peeking phase.

2. If $|D \cap \{k+2, \ldots, m\}| < m - t$, then $J = (\{1, \ldots, k+1\} \setminus D) \cup \{k+2\}$. In this case, let $\alpha = |D \cap \{1, \ldots, k+1\}|$, therefore, $o_J = (k+1-\alpha)+(m-t) = 2t-m+1-\alpha+m-t = t+1-\alpha$. The set $Q_J$ contains at most $t - \alpha < o_J$ corrupted parties, thus, these parties do not get the bit $\sigma_J^{i-1}$ from the trusted party.

Thus, given $\mathrm{PT} =$ “before”, the output of the honest parties is independent of the view of the adversary, hence,

$$\begin{aligned}
&\Pr[V_{\mathrm{REAL}} = v \quad \wedge \quad C_{\mathrm{REAL}} = c \mid \quad \mathrm{PT} = \text{“before”}] \\
&= \Pr[V_{\mathrm{REAL}} = v \mid \quad \mathrm{PT} = \text{“before”}] \cdot \Pr[C_{\mathrm{REAL}} = c \mid \quad \mathrm{PT} = \text{“before”}] \\
&= \Pr[V_{\mathrm{REAL}} = v \mid \quad \mathrm{PT} = \text{“before”}] \cdot \frac{1}{2}.
\end{aligned}$$

In the ideal-world execution, all the messages sent by the simulator in any round $i < i^*$ are independent

of $w_S$, which was the value given to $\mathcal{S}$ by $T_{\text{CoinToss}}$. Thus,

$$\begin{aligned}
&\Pr[V_{\text{IDEAL}} = v \quad \wedge \quad C_{\text{IDEAL}} = c \mid \quad \text{PT} = \text{“before”}] \\
&= \Pr[V_{\text{IDEAL}} = v \mid \quad \text{PT} = \text{“before”}] \cdot \Pr[C_{\text{IDEAL}} = c \mid \quad \text{PT} = \text{“before”}] \\
&= \Pr[V_{\text{IDEAL}} = v \mid \quad \text{PT} = \text{“before”}] \cdot \Pr[w_S = c \mid \quad \text{PT} = \text{“before”}] \\
&= \Pr[V_{\text{IDEAL}} = v \mid \quad \text{PT} = \text{“before”}] \cdot \frac{1}{2}.
\end{aligned}$$

As explained above, $\mathcal{S}$ follows the same random process in interacting with $\mathcal{A}$ (before sending the last message in the premature termination) as does $T$ in the real-world execution. Hence,

$$\Pr[V_{\text{REAL}} = v \mid \quad \text{PT} = \text{“before”}] = \Pr[V_{\text{IDEAL}} = v \mid \quad \text{PT} = \text{“before”}],$$

and, thus,

$$\begin{aligned}
\Pr[V_{\text{REAL}} = v \quad \wedge \quad C_{\text{REAL}} \quad = \quad c \mid \quad \text{PT} = \text{“before”}] = \\
\Pr[V_{\text{IDEAL}} \quad = \quad v \quad \wedge \quad C_{\text{IDEAL}} = c \mid \quad \text{PT} = \text{“before”}].
\end{aligned} \tag{1}$$

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

$$\Pr[V_{\text{REAL}} = v \mid \quad \text{PT} = \text{“after”}] = \Pr[V_{\text{IDEAL}} = v \mid \quad \text{PT} = \text{“after”}],$$

and, thus,

$$\begin{aligned}
\Pr[V_{\text{REAL}} = v \quad \wedge \quad C_{\text{REAL}} \quad = \quad c \mid \quad \text{PT} = \text{“after”}] = \\
\Pr[V_{\text{IDEAL}} \quad = \quad v \quad \wedge \quad C_{\text{IDEAL}} = c \mid \quad \text{PT} = \text{“after”}].
\end{aligned} \tag{2}$$

**$\text{PT} =$ “during” – premature termination occurs in round $i^*$.** This is the interesting case which causes the statistical distance. In the real world, the output of the honest parties is $\sigma_J^{i^*-1}$ for some $J$, while in the ideal world their output is $w_S$. In the first case the output is independent of the adversary's view, while in the second case, the view determining the output.

In both scenarios, premature termination occurs in round $i^*$ and all the bits $\sigma_J^{i^*}$ given to the adversary in round $i^*$ must be the same (otherwise both probabilities are zero). If, however, $c$ does not equal $\sigma_J^{i^*}$ that the adversary sees according to $v$, then we have,

$$\begin{aligned}
\Pr[V_{\text{REAL}} \quad &= \quad v \quad \wedge \quad C_{\text{REAL}} = c \mid \quad \text{PT} = \text{“during”}] \\
&= \quad \Pr[V_{\text{REAL}} = v \mid \quad \text{PT} = \text{“during”}] \cdot \Pr[C_{\text{REAL}} = c \mid \quad \text{PT} = \text{“during”}] \\
&= \quad \Pr[V_{\text{REAL}} = v \mid \quad \text{PT} = \text{“during”}] \cdot \frac{1}{2}
\end{aligned}$$

and

$$\Pr[V_{\text{IDEAL}} = v \quad \wedge \quad C_{\text{IDEAL}} = c \mid \quad \text{PT} = \text{“during”}] = 0.$$

Otherwise, $v$ is consistent both with $i^*$ and with $c$, and we have,

$$\Pr[V_{\text{REAL}} = v \quad \wedge \quad C_{\text{REAL}} = c \mid \quad \text{PT} = \text{“during”}] = \Pr[V_{\text{REAL}} = v \mid \quad \text{PT} = \text{“during”}] \cdot \frac{1}{2}$$

and

$$\begin{aligned} \Pr[V_{\text{IDEAL}} &= v \quad \wedge \quad C_{\text{IDEAL}} = c \mid \quad \text{PT} = \text{“during”}] \\ &= \Pr[V_{\text{IDEAL}} = v \mid \quad \text{PT} = \text{“during”}] \\ &= \Pr[V_{\text{REAL}} = v \mid \quad \text{PT} = \text{“during”}]. \end{aligned}$$

Hence, for any pair $(v, c)$, it holds that,

$$\begin{aligned} \Big| \Pr[V_{\text{IDEAL}} &= v \quad \wedge \quad C_{\text{IDEAL}} = c \mid \quad \text{PT} = \text{“during”}] \\ &- \Pr[V_{\text{REAL}} = v \quad \wedge \quad C_{\text{REAL}} = c \mid \quad \text{PT} = \text{“during”}] \Big| \\ &\leq \frac{1}{2} \Pr[V_{\text{REAL}} = v \mid \quad \text{PT} = \text{“during”}]. \end{aligned} \tag{3}$$

**Bounding the statistical distance between the global outputs of the real world and ideal world.** We next use the above analysis of the three possible cases to bound the statistical distance between the two random variables describing the global outputs of the ideal and real models. For every pair $(v, c)$, the following holds,

$$\begin{aligned} \Pr[V_{\text{IDEAL}} &= v \quad \wedge \quad C_{\text{IDEAL}} = c] \\ &= \Pr[V_{\text{IDEAL}} = v \quad \wedge \quad C_{\text{IDEAL}} = c \mid \quad \text{PT} = \text{“before”}] \cdot \Pr[\text{PT} = \text{“before”}] \\ &\quad + \Pr[V_{\text{IDEAL}} = v \quad \wedge \quad C_{\text{IDEAL}} = c \mid \quad \text{PT} = \text{“during”}] \cdot \Pr[\text{PT} = \text{“during”}] \\ &\quad + \Pr[V_{\text{IDEAL}} = v \quad \wedge \quad C_{\text{IDEAL}} = c \mid \quad \text{PT} = \text{“after”}] \cdot \Pr[\text{PT} = \text{“after”}]. \end{aligned}$$

Similarly,

$$\begin{aligned} \Pr[V_{\text{REAL}} &= v \quad \wedge \quad C_{\text{REAL}} = c] \\ &= \Pr[V_{\text{REAL}} = v \quad \wedge \quad C_{\text{REAL}} = c \mid \quad \text{PT} = \text{“before”}] \cdot \Pr[\text{PT} = \text{“before”}] \\ &\quad + \Pr[V_{\text{REAL}} = v \quad \wedge \quad C_{\text{REAL}} = c \mid \quad \text{PT} = \text{“during”}] \cdot \Pr[\text{PT} = \text{“during”}] \\ &\quad + \Pr[V_{\text{REAL}} = v \quad \wedge \quad C_{\text{REAL}} = c \mid \quad \text{PT} = \text{“after”}] \cdot \Pr[\text{PT} = \text{“after”}]. \end{aligned}$$

Hence, by (1), (2), and (3), for every pair $(v, c)$, we have,

$$\begin{aligned}
\Big|\Pr[V_{\text{IDEAL}} &= v \quad \wedge \quad C_{\text{IDEAL}} = c] - \Pr[V_{\text{REAL}} = v \quad \wedge \quad C_{\text{REAL}} = c]\Big| \\
&= \Big|\Pr[V_{\text{IDEAL}} = v \quad \wedge \quad C_{\text{IDEAL}} = c \mid \quad \text{PT} = \text{“before”}] \cdot \Pr[\text{PT} = \text{“before”}] \\
&\quad - \Pr[V_{\text{REAL}} = v \quad \wedge \quad C_{\text{REAL}} = c \mid \quad \text{PT} = \text{“before”}] \cdot \Pr[\text{PT} = \text{“before”}] \\
&\quad + \Pr[V_{\text{IDEAL}} = v \quad \wedge \quad C_{\text{IDEAL}} = c \mid \quad \text{PT} = \text{“during”}] \cdot \Pr[\text{PT} = \text{“during”}] \\
&\quad - \Pr[V_{\text{REAL}} = v \quad \wedge \quad C_{\text{REAL}} = c \mid \quad \text{PT} = \text{“during”}] \cdot \Pr[\text{PT} = \text{“during”}] \\
&\quad + \Pr[V_{\text{IDEAL}} = v \quad \wedge \quad C_{\text{IDEAL}} = c \mid \quad \text{PT} = \text{“after”}] \cdot \Pr[\text{PT} = \text{“after”}] \\
&\quad - \Pr[V_{\text{REAL}} = v \quad \wedge \quad C_{\text{REAL}} = c \mid \quad \text{PT} = \text{“after”}] \cdot \Pr[\text{PT} = \text{“after”}]\Big| \\
&\leq \frac{1}{2} \Pr[V_{\text{REAL}} = v \mid \quad \text{PT} = \text{“during”}] \cdot \Pr[\text{PT} = \text{“during”}]. \qquad (4)
\end{aligned}$$

This is true since the probability that premature termination occurs in round $i^*$, i.e., $\Pr[\text{PT} = \text{“during”}]$, is the same in both models (again, since $\mathcal{S}$ follows the same random process in interacting with $\mathcal{A}$, before sending the last message in the premature termination, as does $T$ in the real-world execution.)

Now, we are ready to conclude the proof of Claim 5.3.

*Proof of Claim 5.3.*

$$\begin{aligned}
&\text{SD}(\text{IDEAL}, \text{REAL}) \\
&\leq \frac{1}{2} \sum_{(v,c)} \Big|\Pr[V_{\text{IDEAL}} = v \quad \wedge \quad C_{\text{IDEAL}} = c] - \Pr[V_{\text{REAL}} = v \quad \wedge \quad C_{\text{REAL}} = c]\Big| \\
&\leq \frac{1}{2} \sum_{(v,c)} \frac{1}{2} \Pr[V_{\text{REAL}} = v \mid \quad \text{PT} = \text{“during”}] \cdot \Pr[\text{PT} = \text{“during”}] \qquad \text{by (4)} \\
&\leq \frac{1}{2} \Pr[\text{PT} = \text{“during”}] \cdot \sum_{(v,c)} \frac{1}{2} \Pr[V_{\text{REAL}} = v \mid \quad \text{PT} = \text{“during”}] \\
&\leq \frac{1}{2} \Pr[\text{PT} = \text{“during”}].
\end{aligned}$$

The last inequality is due to the properties of the probability function and the fact that $c \in \{0, 1\}$. □

Hence, the statistical distance is bounded by the probability of the adversary guessing $i^*$ correctly (before the abort phase of round $i^*$). We next bound this probability by first showing that the view of the adversary in each round prior to round $i^*$ has probability $1/2^{2^{k+1}}$ to be indistinguishable from $i^*$, and then by showing that this implies that the overall probability for any adversary to guess $i^*$ is at most $2^{2^{k+1}}/r$.

If $\sigma_J^i \neq \sigma_{J'}^i$ for two bits that the adversary receives during the peeking phase of round $i$, then it can learn that $i < i^*$. The information that the adversary obtains in each round are the bits $\sigma_J^i$ for $J$'s such that there are enough corrupted parties in $Q_J$ (i.e., for which $|B \cap Q_J| \geq o_J$). We next claim that the adversary only receives half of the $2^{k+2}$ bits.

We remark that to obtain a total bias of $O(2^{2^{k+2}}/r)$ (instead of the bound of $O(2^{2^{k+1}}/r)$ given in Theorem 5), one may omit the following claim and use the trivial bound of $2^{k+2}$ on the number of bits that the adversary sees in each round. The rest of our proof is independent of this claim. Hence, the reader may choose to skip the technical details of the proof of Claim 5.4 below.

**Claim 5.4.** *Let $B \subset P$ such that $|B| = t < 2m/3$ be the subset of corrupted parties. There exists at most $2^{k+1}$ different subsets $J \subseteq \{1, \ldots, k+2\}$ such that $|B \cap Q_J| \geq o_J$ (i.e., sets that the adversary receives the bits of the associated $Q_J$ in the peeking phase of any round).*

*Proof.* Recall that $Q_{\{j\}} = P_j$ is a singleton (and $o_{\{j\}} = o_j = 1$) for all $1, \ldots, k+1$, $Q_{\{k+2\}} = P_{k+2} = \{p_{k+2}, \ldots, p_m\}$ (and $o_{\{k+2\}} = o_{k+2} = m - t$), and $Q_J = \cup_{j \in J} P_j$. We consider two cases regarding the size of the intersection between the set of corrupted parties $B$ and $Q_{\{k+2\}}$:

1. $\boldsymbol{|B \cap Q_{\{k+2\}}| < o_{\{k+2\}} = m - t.}$ Let $J' \subseteq \{1, \ldots, k+1\}$ be any subset and let $J = J' \cup \{k+2\}$. Thus, $o_J = o_{\{k+2\}} + o_{J'} = m - t + |J'| > |B \cap Q_{\{k+2\}}| + |B \cap Q_{J'}| = |B \cap Q_J|$, since $|B \cap Q_{J'}| \leq |Q_{J'}| = |J'|$. In other words, every such set $Q_J$ contains fewer than $o_J$ corrupted parties. Hence, there are at least $2^{k+1}$ subsets $J$ such that $o_J > |B \cap Q_J|$.

2. $\boldsymbol{|B \cap Q_{\{k+2\}}| \geq o_{\{k+2\}}.}$ In this case, $|B \cap Q_{\{k+2\}}| = m - t + d$ for some $0 \leq d \leq k$. Thus, there are $t - (m - t + d) = 2t - m - d = k - d$ corrupted parties in $\{p_1, \ldots, p_{k+1}\}$, and, therefore, there are $d+1$ honest parties in $\{p_1, \ldots, p_{k+1}\}$. Without loss of generality, we assume that $p_1, \ldots, p_{d+1}$ are honest.

   We argue that there are at least $2^{k+1}$ bits that the adversary does not see by presenting two types of subsets whose bits the adversary cannot see.

   (a) Subsets $J$ such that, $J \subseteq \{1, \ldots, k+1\}$. In this case, $o_J = |Q_J|$ and the adversary can only see that bits of subsets $Q_J$ such that $J \subseteq \{d+2, \ldots, k+1\}$, i.e., $2^{k-d}$ bits. That is, there are $2^{k-d}$ bits that the adversary can see for sets $J \subseteq \{1, \ldots, k+1\}$.

   (b) Subsets $J$, such that $k+2 \in J$. In this case, $o_J = |J \cap \{1, \ldots, k+1\}| + o_{k+2} = |J| - 1 + m - t$. If $\{1, \ldots, d+1\} \subset J$ (where $p_1, \ldots, p_{d+1}$ are honest), then $Q_J$ contains at most $(|J| - 1) - (d+1)$ corrupted parties from $p_1, \ldots, p_{k+1}$ and $m - t + d$ corrupted parties from $\{p_{k+2}, \ldots, p_m\}$. That is, $Q_J$ contains $|J| - 2 + m - t < o_J$ corrupted parties, and the adversary does not know $\sigma_J^{i-1}$.

   There are $2^{k-d}$ sets $J$ that contain $\{1, \ldots, d+1\} \cup \{k+2\}$; for these sets the adversary cannot see the bits. Thus, the adversary sees at most $2^{k+1} - 2^{k-d}$ bits for sets $J$ that contain $\{k+2\}$.

Altogether, there are at least $2^{k-d} + (2^{k+1} - 2^{k-d}) = 2^{k+1}$ bits that the adversary does not see and the adversary sees at most $2^{k+1}$ bits. □

Hence, for each $i < i^*$, with probability $1/2^{2^{k+1}}$ the bits that are available to the adversary are all equal, and the adversary cannot distinguish this round from $i^*$. It is left to show that this implies that the adversary can guess $i^*$ with probability at most $2^{2^{k+1}}/r$. Lemma 2 in [17] proves a generalization of this fact. For the sake of completeness, we include here a proof for our special case.

**Claim 5.5.** *Let $\mathcal{A}$ be an adversary in Protocol $\mathrm{CTWithD}_r$ such that, in each round $i$, there are $\alpha$ bits of the form $\sigma_J^i$ that $\mathcal{A}$ sees. The probability that $\mathcal{A}$ guess $i^*$ is at most $2^{\alpha-1}/r$.*

*Proof.* First, recall that $\mathcal{A}$ is deterministic. In addition, the exact sequence of aborts by corrupted parties is immaterial. Thus, for the rest of this proof, whenever we say that $\mathcal{A}$ aborts in round $i$ we mean that $\mathcal{A}$ instructs a large enough subset of corrupted parties to abort the computation after having seen the bits of round $i$ or in prior rounds (and at least one party in round $i$).

We denote by $\mathcal{V}$ the set of all possible views of $\mathcal{A}$ ending with a premature termination. Clearly,

$$\sum_{v \in \mathcal{V}} \Pr[V_{\text{REAL}} = v \mid i^* = r] \leq 1. \tag{5}$$

For $1 \leq i \leq r$ let $\mathcal{V}_i \subseteq \mathcal{V}$ be the set of all views for which the deterministic $\mathcal{A}$ aborts in round $i$. For a view $v \in \mathcal{V}_i$ denote by $v_{i-1}$ the prefix of $v$ that $\mathcal{A}$ sees until round $i-1$ and denote by $\gamma_i$ the bits that $\mathcal{A}$ sees in the peeking phase of round $i$ (i.e., $v = v_{i-1} \circ \gamma_i$). Without loss of generality, we assume that for any $v \in \mathcal{V}_i$ all the bits that the adversary sees in round $i$ (i.e., $\gamma_i$) are all the same (otherwise, there exists a more successful adversary that never guesses $i^*$ with this view, because if the bits are not the same, then $i \neq i^*$). For any round $i$, let $V^i_{\text{REAL}}$ be a random variable describing the view of the adversary until the end of round $i$.

We next consider the probabilities of the above views when $i^* = i$ and when $i^* = r$. First, note that until round $i$ both views happen with exactly the same probability, i.e.,

$$\Pr[V^{i-1}_{\text{REAL}} = v_{i-1} \mid i^* = i] = \Pr[V^{i-1}_{\text{REAL}} = v_{i-1} \mid i^* = r]. \tag{6}$$

When $i^* = i$, given the prefix of the view, there are only two possible events for the bits of round $i$ each occurring with probability $1/2$, that is, the bits the adversary sees are either all 1 or they are all 0. On the other hand, if $i^* > i$ (specifically, if $i^* = r$), then each of these events happens with probability $1/2^\alpha$. Hence, we have for every $v \in \mathcal{V}_i$ that

$$\Pr[V_{\text{REAL}} = v \mid V^{i-1}_{\text{REAL}} = v_{i-1} \wedge i^* = i] = 2^{\alpha-1} \cdot \Pr[V_{\text{REAL}} = v \mid V^{i-1}_{\text{REAL}} = v_{i-1} \wedge i^* = r]. \tag{7}$$

Combining Equation (6) and Equation (7) we get for every $v \in \mathcal{V}_i$

$$\begin{aligned}
&\Pr[V_{\text{REAL}} = v \mid i^* = i] \\
&= \Pr[V_{\text{REAL}} = v \mid V^{i-1}_{\text{REAL}} = v_{i-1} \wedge i^* = i] \cdot \Pr[V^{i-1}_{\text{REAL}} = v_{i-1} \mid i^* = i] \\
&= 2^{\alpha-1} \cdot \Pr[V_{\text{REAL}} = v \mid V^{i-1}_{\text{REAL}} = v_{i-1} \wedge i^* = r] \cdot \Pr[V^{i-1}_{\text{REAL}} = v_{i-1} \mid i^* = r] \\
&= 2^{\alpha-1} \cdot \Pr[V_{\text{REAL}} = v \mid i^* = r].
\end{aligned}$$

Finally, the probability that $\mathcal{A}$ aborts in $i^*$ is

$$\begin{aligned}
\Pr[\text{PT} = \text{“during”}] &= \sum_{i=1}^{r} \Pr[i = i^*] \Pr[\text{PT} = \text{“during”} \mid i^* = i] \\
&= \sum_{i=1}^{r} \Pr[i = i^*] \sum_{v \in \mathcal{V}_i} \Pr[V_{\text{REAL}} = v \mid i^* = i] \\
&= \frac{1}{r} \sum_{i=1}^{r} \sum_{v \in \mathcal{V}_i} 2^{\alpha-1} \cdot \Pr[V_{\text{REAL}} = v \mid i^* = r] \\
&= \frac{2^{\alpha-1}}{r} \sum_{v \in \mathcal{V}} \Pr[V_{\text{REAL}} = v \mid i^* = r] \leq \frac{2^{\alpha-1}}{r} \qquad \text{by (5).}
\end{aligned}$$

□

**Concluding the proof of Lemma 5.2.** By Claim 5.3 the statistical distance between the global outputs of an ideal-world simulation and a real-world execution of the protocol is bounded by the probability that the adversary guesses the special round $i^*$ in time. In Claim 5.5 we upper bound this probability by $2^\alpha/r$, where $\alpha$ is the number of bits that the adversary sees in each round of the protocol. By Claim 5.4, $\alpha \leq 2^{k+1}$, thus, the statistical distance between the two random variables is upper bounded by $2^{2^{k+1}}/r$ and

$$\mathrm{SD}\left(\mathrm{IDEAL}_{\mathrm{CoinToss}(),\mathcal{S}(\mathrm{aux})}(1^n), \mathrm{REAL}_{\mathrm{CTWithD}_r,\mathcal{A}(\mathrm{aux})}(1^n)\right) \leq 2^{2^{k+1}}/r.$$

□

# 6 Omitting the On-Line Dealer

## 6.1 The Protocol Without the Dealer

In this section we show how Protocol $\mathrm{CTWithD}_r$, presented in Section 5, can be transformed into a real-world protocol. That is, we present a fully-secure $m$-party protocol implementing the ideal functionality described in Protocol $\mathrm{CTWithD}_r$. The resulting protocol has $r'$ rounds, where $r' = r + c(k+1)$, for some constant $c$. The protocol is executed in a network where the parties communicate via an authenticated broadcast channel.

In Figure 4 we describe the initialization functionality, Functionality $\mathrm{MultiShareGen}_r$, which chooses $i^*$, prepares bits for each set for each round, shares those bits, and signs them. In Figure 5, we present Protocol $\mathrm{CoinToss}_r$, which proceeds in rounds and emulates Protocol $\mathrm{CTWithD}_r$. Protocol $\mathrm{CoinToss}_r$ uses a Functionality $\mathrm{Reconstruction}$, described in Figure 6, which reconstructs the output bit if many corrupted parties have aborted. Before formally describing our construction, we outline its main components.

**The inner secret-sharing scheme.** To implement the "ideal secret-sharing functionality" of the trusted party $T$ in Protocol $\mathrm{CTWithD}_r$, we use an $o_J$-out-of-$|Q_J|$ Shamir secret-sharing scheme to share the bits $\sigma_J^i$. That is, in each round $i$, each party $p_j \in Q_J$ obtains a share $S_j^{i,J}$ in an $o_J$-out-of-$|Q_J|$ secret sharing of $\sigma_J^i$. The same requirement on $\sigma_J^i$ as in the protocol with the on-line protocol are preserved using this inner secret-sharing scheme. That is, the adversary is able to obtain information on $\sigma_J^i$ only if it controls at least $o_J$ of the parties in $Q_J$. On the other hand, if, in a premature termination in round $i$, at least $o_J$ parties in $Q_J$ cooperate, then they can reconstruct $\sigma_J^{i-1}$ from their shares.

**The outer secret-sharing scheme.** In the protocol with the on-line dealer, the adversary never learns anything about the bits $\sigma_J^i$ before round $i$ begins. To achieve this property in the real-world protocol, the shares of the inner secret-sharing schemes of all rounds are shared, in a preliminary phase, using a $(t+1)$-out-of-$m$ secret-sharing scheme. The $t+1$ threshold guarantees that the adversary, controlling at most $t$ parties, cannot see the shares of the inner secret-sharing scheme for a given round $i$ without the honest parties, which will not occur before round $i$.

In each round $i$ the parties send messages so that each party can reconstruct its shares in the inner secret-sharing schemes of round $i$. Since all messages are broadcast and all parties can see them, the shares that party $p_j$ receives in round $i$ are masked by using yet another layer of secret sharing. Specifically, a share $S_j^{i,J}$ to be reconstructed by $p_j$ in round $i$ is signed and shared (already in the preliminary phase) in a 2-out-of-2 secret-sharing scheme, such that one share is given to $p_j$ and the other is shared among all parties in a $t$-out-of-$m$ secret-sharing scheme; for details see Construction 2.8. We refer to the combination of these two layers of secret sharing as the outer secret-sharing scheme.

**Premature Termination.** The $t+1$ threshold of the outer secret-sharing scheme enables reconstruction of the shares of the inner scheme as long as at least $t+1$ parties participate in the reconstruction. This allows the real-world protocol to proceed with normal interaction rounds as long as fewer than $m-t$ parties have aborted (as does the ideal-world protocol). If during round $i$ the number of parties that have aborted is at least $m-t$, then an honest majority is guaranteed (since $t < 2m/3$). Thus, in a premature termination in round $i > 1$, the active parties can engage in a fully-secure multiparty computation of a reconstruction functionality.

**Signatures.** In order to confine adversarial strategies to premature aborts, the messages that the parties send are signed (together with the appropriate round number and the index of the sending party), and a verification key is given to all parties. Furthermore, all shares in the inner secret-sharing scheme are signed (as they are used as messages if reconstruction is required). Any message failing to comply with the prescribed protocol is considered an abort message. Since all messages are publicly broadcast, all parties can keep record of all aborts.

**The preliminary phase.** The goal of the preliminary phase is to compute the $\mathrm{MultiShareGen}_r$ functionality, which computes the bits for the underlying sets and the signed shares for the inner and outer secret-sharing schemes. As an honest majority is not guaranteed, it is not possible to implement this functionality by a secure protocol with fairness. That is, we cannot implement an ideal functionality where a trusted party computes the $\mathrm{MultiShareGen}_r$ functionality and sends the appropriate output to each party. However, since the outputs of the $\mathrm{MultiShareGen}_r$ functionality do not reveal any information regarding the output of the protocol to any subset of size at most $t$, fairness is not essential for this part of the computation. We use a protocol with cheat detection, that is, if the protocols fails, then at least one corrupted party is identified by all honest parties. The computation is then repeated without the detected corrupted parties.

More formally, we compute the $\mathrm{MultiShareGen}_r$ functionality using a multiparty protocol that is secure-with-abort and cheat-detection. Informally, this means that we use a protocol that implements the following ideal model: the trusted party computes the $\mathrm{MultiShareGen}_r$ functionality and gives the outputs of the corrupted parties to the adversary; the adversary either sends "proceed", in this case, the trusted party sends the appropriate output to each honest party; otherwise, the adversary sends $\{\text{“}\texttt{abort}_j\text{”}\}_{j\in J}$ for some non-empty set of indeces $J$ of corrupted parties to the trusted party. The trusted party, in turn sends the set $J$ to the honest parties. Using methods from Pass [23], one can obtain a constant-round multiparty protocol secure-with-abort and cheat-detection. See details in Appendix A. Since this protocol is repeated at most $k+1$ times before an honest majority is guaranteed, the round complexity of the preliminary phase is $O(k)$.

### 6.1.1 Proving the Feasibility of the Reconstruction

We next claim that Functionality $\mathrm{Reconstruction}$ described in Figure 6 is well-defined, that is, if the functionality is computed (after premature termination in round $i > 1$), then, indeed, $\sigma_J^{i-1}$ can be reconstructed. To see this, observe that the number of parties in the appropriate set $Q_J$ that participate in the computation (i.e., not in $D$) is at least the reconstruction threshold $o_J$.

**Claim 6.1.** *In any execution of the protocol, there are at least $o_J$ active parties in $Q_J$.*

*Proof.* Let $D$ be the set of aborted parties and $J$ as defined in Functionality $\mathrm{Reconstruction}$.

- If $|D \cap \{k+2,\ldots,m\}| \geq m-t$, then $Q_J = \{p_j : j \in \{1,\ldots,k\} \setminus D\}$ contains only active parties and $|Q_J| = o_J$. Notice that we already proved that in this case $|Q_J| \geq 1$.

**Functionality** $\mathrm{MultiShareGen}_r$

**Notation:** $1^n$ – the security parameter, $r = r(n)$ – the number of rounds in the protocol, $m = m(n)$ – the number of parties, $t = t(n)$ – a bound on the number of corrupted parties, and $k = 2t - m$ – the difference between the number of corrupted parties and the number of honest parties.

**Underlying Subsets:** Let $P_j = \{p_j\}$ for $1 \leq j \leq k+1$ and $P_{k+2} = \{p_{k+2}, \ldots, p_m\}$.
Define $Q_J = \cup_{j \in J} P_j$ for each $J \subset \{1, \ldots, k+2\}$.

For each subset $P_j$ define a reconstruction threshold value $o_j$: For $1 \leq j \leq k+1$ define $o_j = 1$ and define $o_{k+2} = m - t$. Finally, define $o_J = \sum_{j \in J} o_j$ for each $J \subset \{1, \ldots, k+2\}$.

**Computing default bits and signing keys**

1. Choose $w \in \{0,1\}$ and $i^* \in \{1, \ldots, r\}$ uniformly at random.
2. For each $i \in \{1, \ldots, r\}$ and for each $J \subset \{1, \ldots, k+2\}$,
   (a) if $i \leq i^* - 1$, then choose independently and uniformly at random $\sigma_J^i \in \{0,1\}$.
   (b) if $i \geq i^*$, then set $\sigma_J^i = w$.
3. Compute $(K_{\mathrm{sign}}, K_{\mathrm{ver}}) \leftarrow \mathrm{Gen}(1^n)$.

**Computing signed shares of the inner secret-sharing scheme**

4. For each $i \in \{1, \ldots, r\}$ and for each $J \subset \{1, \ldots, k+2\}$
   (a) Share $\sigma_J^i$ using an $o_J$-out-of-$|Q_J|$ Shamir's secret-sharing scheme for the parties in $Q_J$. For each party $p_j \in Q_J$, let $S_j^{i,J}$ be its share of $\sigma_J^i$.
   (b) Sign each share $S_j^{i,J}$: $R_j^{i,J} \leftarrow (S_j^{i,J}, i, J, j, \mathrm{Sign}((S_j^{i,J}, i, J, j), K_{\mathrm{sign}})$.

**Computing shares of the outer secret-sharing scheme**

5. For each $i \in \{1, \ldots, r\}$, for each $J \subset \{1, \ldots, k+2\}$, and for each $p_j \in Q_J$: share $p_j$'s signed share $R_j^{i,J}$ using a $(t+1)$-out-of-$m$ secret-sharing scheme with respect to $p_j$ as defined in Construction 2.8; that is, one masking share $\mathrm{mask}_j(R_j^{i,J})$ and $m-1$ complement shares $(\mathrm{comp}_1(R_j^{i,J}), \ldots, \mathrm{comp}_{j-1}(R_j^{i,J}), \mathrm{comp}_{j+1}(R_j^{i,J}), \ldots, \mathrm{comp}_m(R_j^{i,J}))$ are produced.

**Signing the messages of all parties**

6. For every $1 \leq q \leq m$, compute the message $m_{(q,i)}$ that $p_q \in P$ broadcasts in round $i$ by concatenating (1) $q$, (2) the round number $i$, and (3) the complement shares $\mathrm{comp}_q(R_j^{i,J})$ for all $J$ and for all $j \neq q$ such that $p_j \in Q_J$ produced in Step (5) for $p_q$.
7. Compute $M_{q,i} \leftarrow (m_{q,i}, \mathrm{Sign}(m_{q,i}, K_{\mathrm{sign}}))$.

**Outputs:** Each party $p_j$ receives

- The verification key $K_{\mathrm{ver}}$.
- The messages $M_{j,1}, \ldots, M_{j,r}$ that $p_j$ broadcasts during the protocol.
- $p_j$'s private masks $\mathrm{mask}_j(R_j^{i,J})$ which were produced in Step (5) for each $1 \leq i \leq r$ and each $J \subset \{1, \ldots, k+2\}$ such that $p_j \in Q_J$.

Figure 4: The initialization functionality $\mathrm{MultiShareGen}_r$.

**An $m$-party protocol $\mathrm{CoinToss}_r$**

**Notations:** $1^n$ – the security parameter, $r = r(n)$ – the number of rounds in the protocol, $m = m(n)$ – the number of parties, $t = t(n)$ – a bound on the number of corrupted parties, and $k = 2t - m$ – the difference between the number of corrupted parties and the number of honest parties.

**Joint input:** Security parameter $1^n$.

**Preliminary phase:**

1. $D \leftarrow \emptyset$
2. If $|D| < m - t$,
   (a) The parties execute a secure-with-abort and cheat-detection protocol computing Functionality $\mathrm{ShareGenWithAbort}_r$.
   (b) If a set of parties aborts, that is, the output of the honest parties is $\{\text{“}\texttt{abort}_j\text{”}\}_{j \in J}$ for some non-empty set $J$, then $D \leftarrow D \cup J$ and repeat Step (2) without the parties whose indices are in $D$.
3. Else, if $|D| \geq m - t$, then the premature termination step is executed with $i = 1$.

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

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

**Premature termination step**

6. If $i = 1$, then the active parties (parties $p_j$ such that $j \in D$) use a multiparty secure protocol (with fairness) to toss a fair coin, output the resulting bit, and halt.
7. Otherwise,
   (a) Each party $p_j$ reconstructs $R_j^{i-1,J}$, the signed share of the inner secret-sharing scheme produced in Step (4) of Functionality $\mathrm{MultiShareGen}_r$, for every $J \subset \{1, \ldots, k+2\}$ such that $p_j \in Q_J$.
   (b) The active parties execute a secure multiparty protocol (with an honest majority) computing Functionality $\mathrm{Reconstruction}$, where the input of each party $p_j$ is $R_j^{i-1,J}$ for every $J \subset \{1, \ldots, k+2\}$ such that $p_j \in Q_J$.
   (c) The active parties return the output of this protocol, and halt.

**At the end of round $r$:**

8. Each active party $p_j$ broadcasts the signed shares $R_j^{r,J}$ for each $J$ such that $p_j \in Q_J$.
9. Let $J$ be the lexicographically first set such that at least $o_J$ parties broadcast properly signed shares $R_j^{r,J}$. Each active party reconstructs the bit $\sigma_J^r$ of $J$, outputs $\sigma_J^r$, and halts.

Figure 5: Protocol $\mathrm{CoinToss}_r$.

**Functionality** Reconstruction

**Notations:** $1^n$ – the security parameter, $m = m(n)$ – the number of parties, $t = t(n)$ – a bound on the number of corrupted parties, and $k = 2t - m$ – the difference between the number of corrupted parties and the number of honest parties.

**Joint Input:** The indices of inactive parties, $D$, and the verification key, $K_{\text{ver}}$.

**Private Input of $p_j$:** A set of signed shares $R_j^{i-1,J}$ for each $J \subset \{1, \ldots, k+2\}$ such that $p_j \in Q_J$.

**Computation:**

1. For each $p_j$, if $p_j$'s input is not appropriately signed or malformed, then $D \leftarrow D \cup \{j\}$.
2. Define the set $J$:
   - If $|D \cap \{k+2, \ldots, m\}| \geq m - t$ then $J = \{1, \ldots, k+1\} \setminus D$.
   - If $|D \cap \{k+2, \ldots, m\}| < m - t$ then $J = (\{1, \ldots, k+1\} \setminus D) \cup \{k+2\}$.
3. Reconstruct $\sigma_J^{i-1}$ from the shares of the active parties in $Q_J$.

**Outputs:** Each honest party $p_j$ outputs the value $\sigma_J^{i-1}$, the secret reconstructed from the signed shares $R_j^{i-1,J}$.

Figure 6: The functionality for reconstruction the output bit if the premature termination step is executed in round $i > 1$ .

- If $|D \cap \{k+2, \ldots, m\}| \leq m - t - 1$, then $o_J = |J| - 1 + m - t$ and $|Q_J \setminus \{p_j : j \in D\}| = |J| - 1 + |\{k+2, \ldots, m\} \setminus D|$. To prove that $|Q_J \setminus \{p_j : j \in D\}| \geq o_J$, it suffices to show that $|\{k+2, \ldots, m\} \setminus D| \geq m - t$:

$$|\{k+2, \ldots, m\} \setminus D| \geq (m - k - 1) - (m - t - 1) = t - k = t - (2t - m) = m - t.$$

□

## 6.2 The Simulator for the Protocol Without a Dealer

We next prove the security of Protocol $\text{CoinToss}_r$ and, hence, the correctness of Theorem 6. As explained in Section 4 we prove that this protocol is a real-world implementation of the (ideal) functionality of Protocol $\text{CTWithD}_r$, the protocol with the on-line dealer. We analyze Protocol $\text{CoinToss}_r$ in a hybrid model with the following 3 ideal implementations of the functionalities that we described before:

**$\text{ShareGenWithAbort}_r$.** This is an implementation of Functionality $\text{MultiShareGen}_r$ in a secure-with-abort and cheat-detection model. That is, first Functionality $\text{MultiShareGen}_r$ is executed. Then, the adversary gets the outputs of the corrupted parties of the functionality. Next, the adversary decides whether to halt or to continue; If the adversary decides to continue, it sends a "proceed" message to Functionality $\text{ShareGenWithAbort}_r$, which sends the honest parties their outputs. Otherwise, the adversary sends $\{\text{“abort}_j\text{”}\}_{j \in J}$ for some set of corrupted parties indexed by $J$, and this message is sent to the honest parties.

**FairCoinToss.** This is an implementation with full security (with fairness) of the functionality that computes a uniformly distributed random bit. That is, the functionality simply chooses a random bit and gives it to all parties.

**Reconstruction.** This is an implementation with full security (with fairness) of the functionality described in Figure 6; this functionality is used in the premature termination step in Protocol $\mathrm{CoinToss}_r$ for reconstructing the output bit from the shares of the previous round.

When the last two functionalities are executed, an honest majority is guaranteed, hence, these functionalities can be implemented with full security.

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

**Simulating the preliminary phase:** Upon receiving a "start" message from the ideal-world trusted party $T_{\mathrm{CTWithD}}$, the simulator $\mathcal{S}$ plays the role of Functionality $\mathrm{ShareGenWithAbort}_r$ (possibly, up to $m - t$ times), simulating the interaction between the adversary $\mathcal{A}$ and this functionality.

1. $D \leftarrow \emptyset$.
2. The simulator $\mathcal{S}$ prepares outputs for the corrupted parties in a single interaction with Functionality $\mathrm{ShareGenWithAbort}_r$. The simulator $\mathcal{S}$ does this by first setting $\sigma_J^i = 1$, for all $J \subset \{1, \ldots, k+2\}$ and for all $i \in \{1, \ldots, r\}$. Then, $\mathcal{S}$ follows Steps 3–7 in the computation of Functionality $\mathrm{MultiShareGen}_r$ (skipping Step (1) and Step (2)) to obtain uniformly distributed shares for the parties.[3] The simulator $\mathcal{S}$ keeps the key $K_{\mathrm{sign}}$ for future use.
3. For each $p_j \in B$, the simulator $\mathcal{S}$ sends to $\mathcal{A}$:
   - The verification key $K_{\mathrm{ver}}$.
   - The masking shares $\mathrm{mask}_j\,(R_j^{i,J})$ for each $i \in \{1, \ldots, r\}$ and for each $J \subset \{1, \ldots, k+2\}$ such that $p_j \in Q_J$.
   - The messages $M_{j,1}, \ldots, M_{j,r}$.
4. For each $p_j \in B$, if $\mathcal{A}$ sends an "$\mathtt{abort}_j$" message behalf of $p_j$ to $\mathcal{S}$, then $\mathcal{S}$ sends "$\mathtt{abort}_j$" to the trusted party $T_{\mathrm{CTWithD}}$.
   (a) $D \leftarrow D \cup \{j\}$.
   (b) If $|D| < m - t$, then Steps 2–3 are repeated.
   (c) Otherwise ($|D| \geq m - t$), the simulator $\mathcal{S}$ executes the simulation of the premature termination step with $i = 1$.
5. If $\mathcal{A}$ sends a "$\mathtt{continue}$" message to $\mathcal{S}$ (simulating Functionality $\mathrm{ShareGenWithAbort}_r$), then $\mathcal{S}$ sends an "$\mathtt{abort}_j$" message for each party $p_j \in D$ to the trusted party $T_{\mathrm{CTWithD}}$ (and gets in return a "proceed" message). The simulator $\mathcal{S}$ denotes by $B_0 = B \setminus D$ the set of active corrupted parties after the simulation of the execution of $\mathrm{ShareGenWithAbort}_r$ (from here on, $\mathcal{S}$ will interact with parties in $B_0$).

[3] These shares are temporary and will later be opened to the actual bits obtained from $T_{\mathrm{CTWithD}}$ during the interaction rounds using the properties of Shamir's secret-sharing scheme; specifically, they can alternatively be computed by setting $\sigma_J^i$ to 0.

**Simulating interaction rounds:**

Let $\mathcal{J}_{B_0}$ be the collection of subsets $J \subset \{1,\ldots,k+2\}$ such that there are enough corrupted parties in $Q_J$ to reconstruct the bits associated with $J$, i.e., $|B_0 \cap Q_J| \geq o_J$.
To simulate round $i$ for $i = 1,\ldots,r$, the simulator $\mathcal{S}$ proceeds as follows:

1. $\mathcal{S}$ gets from the trusted party $T_{\mathrm{CTWithD}}$ the bits that the corrupted parties see, that is, $\mathcal{S}$ gets a bit $\tau_J^i$ for each $J \in \mathcal{J}_{B_0}$.[4]
2. The simulator $\mathcal{S}$ selects the shares in the inner secret-sharing scheme for corrupted parties: For every $J \in \mathcal{J}_{B_0}$, the simulator $\mathcal{S}$ selects uniformly at random shares of $\tau_J^i$ in an $o_J$-out-of-$|Q_J|$ Shamir secret-sharing scheme. Denote these shares by $\left\{X_j^{i,J} : p_j \in Q_J\right\}$. For each $p_j \in Q_J$, let $Y_j^{i,J} \leftarrow (X_j^{i,J}, i, J, j, \mathrm{Sign}((X_j^{i,J}, i, J, j), K_{\mathrm{sign}}))$.
3. The simulator $\mathcal{S}$ selects complementary shares for all honest parties, with the requirement that, together with the masking share $\mathrm{mask}_j\left(R_j^{i,J}\right)$ held by $p_j$ and the complementary shares held by the corrupted parties in $B_0 \setminus \{p_j\}$, they are a sharing of $Y_j^{i,J}$.
   To do so, for every $J \in \mathcal{J}_{B_0}$ and for each $p_j \in B_0$,
   (a) $\mathcal{S}$ calculates $\alpha_j = \mathrm{mask}_j\left(R_j^{i,J}\right) \oplus Y_j^{i,J}$.
   (b) $\mathcal{S}$ generates $m-t$ shares of $\alpha_j$ uniformly at random from all possible vectors of $m-t$ shares in a $t$-out-of-$(m-1)$ Shamir secret-sharing scheme, where together with the $t$ given shares $\left\{\mathrm{comp}_q\left(R_j^{i,J}\right) : p_q \in B \setminus p_j\right\}$ which were produced in Step (2) in the simulation of the preliminary phase, they will form a secret sharing of $Y_j^{i,J}$.
   (This is possible according to Construction 2.8, since the shares held by the corrupted parties before round $i$ do not determine the reconstructed value).
   Denote by $\mathrm{comp}_q\left(Y_j^{i,J}\right)$ the complementary share that $\mathcal{S}$ generates for the honest party $p_q$ for $p_j \in B_0 \cap Q_J$, where $J \in \mathcal{J}_{B_0}$.
4. For party $p_j$ and subset $J$, such that, either $p_j \notin B$ or $J \notin \mathcal{J}_{B_0}$, let $\mathrm{comp}_q\left(R_j^{i,J}\right)$ be the complementary share which was produced in Step (2) in the simulation of the preliminary phase, i.e., $\mathrm{comp}_q\left(R_j^{i,J}\right)$.
5. Construct signed messages for the honest parties: To construct the message $M'_{q,i}$ for an honest party $p_q$ in round $i$, the simulator computes $m'_{q,i}$ by concatenating the following strings:
   (a) $q$,
   (b) The round number $i$,
   (c) The complement shares which were described in Step (4) above,
   (d) The complement shares $\mathrm{comp}_q\left(Y_j^{i,J}\right)$ for all $J \in \mathcal{J}_{B_0}$ and for all $j \neq q$ such that $p_j \in Q_J$ produced in Step (3) for $p_q$.

   Then, $\mathcal{S}$ signs $m'_{q,i}$, i.e., $\mathcal{S}$ computes $M'_{q,i} \leftarrow (m'_{q,i}, \mathrm{Sign}(m'_{q,i}, K_{\mathrm{sign}}))$.
6. The simulator $\mathcal{S}$ sends all the message $M'_{q,i}$ on behalf of each honest party $p_q$ to $\mathcal{A}$.
7. If $\mathcal{A}$ sends an invalid or no message on behalf of $p_j$, then $\mathcal{S}$ sends "$\mathtt{abort}_j$" to $T_{\mathrm{CTWithD}}$.
   (a) $D \leftarrow D \cup \{j\}$.

[4] In Steps 2–5, the simulator $\mathcal{S}$ constructs the messages of the honest parties in order to allow the corrupted parties in each $J \in \mathcal{J}_{B_0}$ to reconstruct $Y_j^{i,J}$.

(b) If $|D| \geq m - t$, then $\mathcal{S}$ executes the simulation of the premature termination step.

(c) Otherwise, the simulator $\mathcal{S}$ proceeds to the next round.

**Simulating the premature termination step:**

- If $i = 1$, $\mathcal{S}$ simulates $\mathcal{A}$'s interaction with Functionality $\mathrm{FairCoinToss}$ as follows:
  1. The simulator $\mathcal{S}$ gets from $T_{\mathrm{CTWithD}}$ the bit $\sigma$. Then, $\mathcal{S}$ sends $\sigma$ to $\mathcal{A}$.
- If $i > 1$, $\mathcal{S}$ simulates $\mathcal{A}$'s interaction with Functionality $\mathrm{Reconstruction}$ as follows:
  1. $\mathcal{S}$ receives from $\mathcal{A}$ the inputs of the active corrupted parties in the current round denoted by $B_i$. If the input for $p_j \in B_i$ is not appropriately signed, then $\mathcal{S}$ sends "$\texttt{abort}_j$" to $T_{\mathrm{CTWithD}}$.
  2. $\mathcal{S}$ gets from $T_{\mathrm{CTWithD}}$ a bit $\sigma$ and sends it to $\mathcal{A}$.

**Simulating normal termination at the end of round $r$:**

1. The simulator gets $w$ from the trusted party $T_{\mathrm{CTWithD}}$.
2. $\mathcal{S}$ constructs all the singed shares of the inner secret-sharing scheme for each $J \subset \{1, \ldots, k+2\}$ and for each honest party $p_j \in Q_J$ as follows.
   - For each $J \in \mathcal{J}_{B_0}$, the signed shares $\left\{Y_j^{r,J}\right\}$ were produced in Step (2) of the simulation of the interaction of round $r$.
   - For each $J \notin \mathcal{J}_{B_0}$, the simulator $\mathcal{S}$ selects uniformly at random secret shares of $w$ in an $o_J$-out-of-$|Q_J|$ Shamir secret-sharing scheme. Denote these shares by $\left\{X_j^{r,J}\right\}$.

     For each share $X_j^{r,J}$, the simulator concatenates the corresponding identifying details, and signs them to obtain: $Y_j^{r,J} \leftarrow (X_j^{r,J}, r, J, j, \mathrm{Sign}((X_j^{r,J}, r, J, j), K_{\mathrm{sign}}))$.

     ($Y_j^{r,J}$ is the share of inner secret-sharing scheme that party $p_j \in Q_J$ should reconstruct.)
3. For each honest party $p_j$, the simulator $\mathcal{S}$ sends to $\mathcal{A}$ the shares $Y_j^{r,J}$ for all subsets $J$, such that $p_j \in Q_J$.

**The Output of the Simulator:** The simulator $\mathcal{S}$ outputs $r_{\mathcal{A}}$ and the sequence of messages exchanged between $\mathcal{S}$ and the adversary $\mathcal{A}$ and halts.

## 6.3 Proof of the Correctness of the Simulation

In this section we prove that the simulator described in Section 6.2 simulates Protocol $\mathrm{CoinToss}_r$ described in Section 6.1. We consider the two random variables from Section 2.1, both of the form $(V, C)$, where $V$ describes a possible view of $\mathcal{A}$, and $C$ describes a possible output of the honest parties (i.e., $C \in \{0,1\}$). The first random variable $\mathrm{REAL}_{\mathrm{CoinToss}_r, \mathcal{A}(\mathrm{aux})}(1^n)$ describes the hybrid world, i.e., an execution of the $\mathrm{CoinToss}_r$ protocol in the above hybrid model, where $V$ describes the view of the adversary $\mathcal{A}$ in this execution, and $C$ is the output of the honest parties in this execution. The second random variable $\mathrm{IDEAL}_{\mathrm{CTWithD}_r, \mathcal{S}(\mathrm{aux})}(1^n)$ describes the output of the simulator and the honest parties in the ideal world, i.e., an execution of the $r$-round $m$-party coin-tossing protocol $\mathrm{CTWithD}_r$ with an on-line dealer, where $V$ describes the output of the simulator $\mathcal{S}$ in this execution, and $C$ is the output of the honest parties in this execution. Our goal here is to show that the these two random variables are computationally indistinguishable. For the rest of this proof, we simplify notations by omitting all notations that are clear from the context; we

denote the above two random variables by $\mathrm{REAL} = (V_{\mathrm{REAL}}, C_{\mathrm{REAL}})$ and $\mathrm{IDEAL} = (V_{\mathrm{IDEAL}}, C_{\mathrm{IDEAL}})$ respectively.

In the main part of the proof, we assume that an adversary never (successfully) forges a signature during the execution of the protocol. Under this assumption, we prove that these two random variables are identically distributed. Since the probability that a polynomial-time adversary successfully forges a signature for any message is negligible, the statistical distance between these two random variables is negligible.

We next prove a slightly stronger claim than the one required for the above. We define two random variables $\mathrm{KEY}_{\mathrm{IDEAL}}$ and $\mathrm{KEY}_{\mathrm{REAL}}$ representing the sequence of signing keys used in the (repeated) iterations of the preliminary phase to sign the messages. That is, $\mathrm{KEY}_{\mathrm{IDEAL}}$ represents the sequence of keys that were used to sign the messages in the computations of Step (2) during a given execution of the simulation, and $\mathrm{KEY}_{\mathrm{REAL}}$ represents the sequence of keys that were used to sign the messages in the computations of Step (2a) (i.e., in the executions of Functionality $\mathrm{ShareGenWithAbort}_r$) in an execution of Protocol $\mathrm{CoinToss}_r$ in the hybrid model. The signing keys are not part of the view of the adversary. However, we prove that the joint distributions $(V_{\mathrm{REAL}}, C_{\mathrm{REAL}}, \mathrm{KEY}_{\mathrm{REAL}})$ and $(V_{\mathrm{IDEAL}}, C_{\mathrm{IDEAL}}, \mathrm{KEY}_{\mathrm{IDEAL}})$ are identically distributed makes the proof easier and at the same time proves that REAL and IDEAL are identically distributed. Let $\mathrm{REAL}_+ = (V_{\mathrm{REAL}}, C_{\mathrm{REAL}}, \mathrm{KEY}_{\mathrm{REAL}})$ and $\mathrm{IDEAL}_+ = (V_{\mathrm{IDEAL}}, C_{\mathrm{IDEAL}}, \mathrm{KEY}_{\mathrm{IDEAL}})$.

We consider $r+2$ random variables $\left\{\mathrm{REAL}_+^i\right\}_{i=0}^{r+1}$ that are related to $\mathrm{REAL}_+$ and $r+2$ random variables $\left\{\mathrm{IDEAL}_+^i\right\}_{i=0}^{r+1}$ that are related to $\mathrm{IDEAL}_+$. Each of these variables describes a partial execution until a given point as next described. The random variable $\mathrm{REAL}_+^0$ describes the execution until the end of the preliminary phase in Protocol $\mathrm{CoinToss}_r$. That is, $\mathrm{REAL}_+^0$ contains the sequence of signing keys used by Functionality $\mathrm{ShareGenWithAbort}_r$ and the messages that the adversary $\mathcal{A}$ exchanges with Functionality $\mathrm{ShareGenWithAbort}_r$ in all interactions with this functionality, i.e., until either $\mathcal{A}$ sends a "`continue`" message to the functionality or until $\mathcal{A}$ interacts with the functionality $m - t$ times. Similarly, the random variable $\mathrm{IDEAL}_+^0$ describes the simulation until the end of all iterations of the simulator $\mathcal{S}$. In addition, $\mathrm{IDEAL}_+^{r+1}$ contains the interaction in the termination step (normal or premature termination).

For each $1 \leq i \leq r$, the random variable $\mathrm{REAL}_+^i$ contains $\mathrm{REAL}_+^{i-1}$ and, in addition, contains the messages that were exchanged between the adversary (i.e., the corrupted parties) and the honest parties in the first $i$ rounds in the execution of Protocol $\mathrm{CoinToss}_r$ (in the hybrid model). Specifically, if the execution was prematurely terminated before round $i$, then we have $\mathrm{REAL}_+^i = \mathrm{REAL}_+^{i-1}$. The random variable $\mathrm{IDEAL}_+^i$ is defined similarly.

We next prove that for every $0 \leq i \leq r+1$, it holds that $\mathrm{REAL}_+^i \equiv \mathrm{IDEAL}_+^i$ (assuming no signatures were forged). Towards this goal, we fix a triple $\lambda = (v, c, \mathrm{keys})$ and show that for every $i$ the probability that $\mathrm{IDEAL}_+^i$ is consistent with $\lambda$ is the same as the probability that $\mathrm{REAL}_+^i$ is consistent with $\lambda$, where in both scenarios we invoke $\mathcal{A}$ on $1^n$ and the same auxiliary information $\mathrm{aux}$.

### 6.3.1 The Initialization Step

**Claim 6.2.** *The probability that* $\mathrm{REAL}_+^0$ *is consistent with* $\lambda$ *is the same as the probability that* $\mathrm{IDEAL}_+^0$ *is consistent with* $\lambda$*.*

*Proof.* The simulator follows the same procedure for generating the messages to corrupted parties as the protocol with one main difference. The simulator chooses $\sigma_J^i = 1$ for every $1 \leq i \leq r$ and $J \subset \{1, \ldots, k+2\}$, while the protocol chooses these values at random. However, since the adversary holds at most $t$ shares of Construction 2.8 with threshold $t+1$ (i.e., $\alpha = t+1$), these shares are uniformly distributed (as explained in Construction 2.8). Thus, the probability that the adversary sees the shares that are consistent with $\lambda$ is the

same in both models. As the messages sent to the corrupted parties are these shares and signatures on these shares, the claim follows. □

### 6.3.2 The Interaction Rounds of the Protocol

**Claim 6.3.** *For every $1 \leq i \leq r$, the following two probabilities are equal:*

1. *The probability that $\mathrm{REAL}^i_+$ is consistent with $\lambda$, given that $\mathrm{REAL}^{i-1}_+$ is consistent with $\lambda$.*
2. *The probability that $\mathrm{IDEAL}^i_+$ is consistent with $\lambda$, given that $\mathrm{IDEAL}^{i-1}_+$ is consistent with $\lambda$.*

*Proof.* If the execution prematurely terminated in round $i' < i$, then there is nothing to prove, since by definition $\mathrm{REAL}^i_+ = \mathrm{REAL}^{i-1}_+$ and $\mathrm{IDEAL}^i_+ = \mathrm{IDEAL}^{i-1}_+$ and, thus, are both consistent with $\lambda$. We, thus, assume that round $i$ does take place according to $\lambda$.

The messages broadcast in round $i$ are shares in the outer secret-sharing schemes of shares generated in an inner secret-sharing scheme. We will show that in both worlds the hybrid and the ideal, they are distributed according to the same distribution. The selection of the bits that the adversary sees throughout the protocol is done by the same process in the real protocol (by Functionality $\mathrm{ShareGenWithAbort}_r$) and in the ideal world (by the trusted dealer). That is, in both cases the bits are selected according to the algorithm defined by Functionality $\mathrm{MultiShareGen}_r$. Furthermore, given these values, each bit is shared independently of the other bits. Thus, it suffices to show that the outer shares of each bits are equally distributed. There are two cases:

**Outer shares for $p_j \in B_0$ and $J \in \mathcal{J}_{B_0}$ such that $j \in J$.** Since $J \in \mathcal{J}_{B_0}$, the simulator gets from the trusted party $T_{\mathrm{CTWithD}}$ a bit $\tau^i_J$. The simulator then generates shares of an inner secret-sharing scheme of $\tau^i_J$, and signs these shares. Let $Y^{i,J}_j$ be the share of $p_j$ generated by the simulator. This share is generated in the same process as the share $R^{i,J}_j$ generated in the hybrid world.

The process of generating the outer shares of $Y^{i,J}_j$ and $R^{i,J}_j$ (in the ideal and hybrid worlds, respectively) are different. In the hybrid world, $R^{i,J}_j$ is simply shared using Construction 2.8. In the ideal world, the values $\left\{\mathrm{comp}_q\left(R^{i,J}_j\right)\right\}_{q\in B_0}$ (generated in the simulation of the preliminary phase) are completed to $m$ shares of Construction 2.8 for the secret $Y^{i,J}_j$. As explain in Construction 2.8, the set of at most $t$ shares of $R^{i,J}_j$ are uniformly distributed, thus, their completion to shares of $Y^{i,J}_j$ is distributed as in the hybrid world.

**Outer shares for $p_j \notin B_0$ or $J \notin \mathcal{J}_{B_0}$.** In the hybrid world, first the value $\sigma^i_J$ is chosen in Functionality $\mathrm{MultiShareGen}_r$, then this bit is shared in an inner secret-sharing scheme, signed, and shared in an outer secret-sharing scheme. In the ideal world, the process is similar, however for the fixed bit $\sigma^i_J = 1$.

- If $p_j \notin B_0$, the parties in $B_0$ miss the mask of $R^{i,J}_j$, thus, they cannot see the difference between the worlds.
- If $J \notin \mathcal{J}_{B_0}$, the number of parties in $B_0$ that are in $Q_J$ is fewer than $o_J$. Thus, the inner shares given to the parties in $Q_J \cap B_0$ are equally distributed regardless of the value of $\sigma^i_J$, and their sharing in the outer secret-sharing scheme is equally distributed in the two worlds.

□

### 6.3.3 The Termination Step

We next prove that the random variables are equally distributed after the termination (premature or normal).

**Claim 6.4.** *The following two probabilities are equal:*

1. *The probability that* $\mathrm{REAL}_+$ *is consistent with* $\lambda$*, given that* $\mathrm{REAL}_+^r$ *is consistent with* $\lambda$*.*
2. *The probability that* $\mathrm{IDEAL}_+$ *is consistent with* $\lambda$*, given that* $\mathrm{IDEAL}_+^r$ *is consistent with* $\lambda$*.*

*Proof.* We need to show that the probabilities that in both worlds, the view of the adversary $\mathcal{A}$ in the termination phase is consistent with $\lambda$, given that everything else was consistent with $\lambda$ is the same in both worlds. We consider the two possible types of termination of the protocol, i.e., normal termination and premature termination. The type of termination we need to consider is specified by the view in previous round, and, thus, by $\mathrm{REAL}_+^r$ and $\mathrm{IDEAL}_+^r$. We therefore consider the type of termination as specified by $\lambda$.

**A premature termination step is specified by $\lambda$:** We separate the analysis of this case into two. First, we consider the case where the premature termination takes place with $i = 1$. Then, we consider the case where $1 < i \leq r$. In all of the cases discussed below, at least $m - t$ parties have aborted, and, thus, an honest majority is guaranteed among the remaining active parties.

1. The case where $i = 1$ (according to $\lambda$): This case is possible either if according to $\lambda$ at least $m - t$ parties aborted in the preliminary phase, or $m - t$ parties aborted before the first round of interaction was completed. By the definition of the premature termination step, in both the above cases the output of the protocol should be a uniform bit that is independent of the view of the adversary. We next claim that both in the hybrid-world and in the simulation this is indeed the case. Formally, we claim that given that the view of the adversary (as well as the sequence of signing keys used thus far) is consistent with $\lambda$, the view of the adversary in the premature termination step will consist of an independent uniform bit, both in the hybrid-world and in the simulation.

   In the premature termination step in the hybrid-world, the parties use Functionality $\mathrm{FairCoinToss}$ to toss a completely fair coin (the functionality is fully secure, since an honest majority is guaranteed). In the simulation, the simulator $\mathcal{S}$ sends an "$\mathtt{abort}_j$" message to $T_{\mathrm{CTWithD}}$ for each party $p_j$ that has aborted. Then, $\mathcal{S}$ gets in return a bit $\sigma$ and sends $\sigma$ to $\mathcal{A}$. By the definition of Protocol $\mathrm{CTWithD}_r$ the trusted party will return $w' = \sigma_J^0$, for some subset $J$. Regardless of the exact subset, this bit is uniform and independent of the view of the adversary.
2. The case where $1 < i \leq r$ (according to $\lambda$): We claim that in this case the output of the honest parties (which is part of the view of the adversary) is the bit $\sigma_J^{i-1}$ for the same subset $J$, both in the hybrid-world and in the simulation. This suffices, since, as explained above, the bits for all sets are selected according to the same randomized process in the real protocol (by Functionality $\mathrm{ShareGenWithAbort}_r$) and in the ideal world (by the trusted dealer). Thus, given that the same view for the adversary.

   In the hybrid-world the parties execute Functionality $\mathrm{Reconstruction}$ to select the appropriate bit (the functionality is fully secure, since an honest majority is guaranteed). In the ideal world the simulator turns to the trusted party $T_{\mathrm{CTWithD}}$ for the output bit. In both cases the bit is chosen according to the set $D$ of parties that have aborted the protocol. By the assumption that the view of $\mathcal{A}$ thus far is consistent with $\lambda$, we have that the set of parties that aborted before the beginning of the premature termination step has started is the same in both models. Now,

all inputs of corrupted party to Functionality $\mathrm{Reconstruction}$ are going to be the same both in the hybrid-world and in the simulation, thus the subset $D$ is going to be the same. Therefore, in both models, the same subset $J$ is going to be selected, and by the above the same bit is going to be output.

**A normal termination step is specified by $\lambda$:** According to specification of Protocol $\mathrm{CTWithD}_r$, at the end of round $r$, each party $p_j$ broadcasts its (signed) shares in the inner secret-sharing scheme of round $r$, i.e., party $p_j$ broadcasts $R_j^{r,J}$ for each $J$ such that $p_j \in Q_J$. Thus, in the hybrid-world, the adversary will receive the shares sent by the honest parties. To simulate the messages sent by the honest parties in this case, the simulator $\mathcal{S}$ gets the output bit $w$ from the trusted party $T_{\mathrm{CTWithD}}$, selects inner secret-sharing schemes shares for the honest parties, and signs them. That is, $\mathcal{S}$ selects for each honest party $p_j$ shares $\{X_j^{r,J}\}$ and then concatenates the corresponding identifying details, and signs them to obtain: $\{Y_j^{r,J}\}$.

We first consider the issue of the output bit itself. If $\mathcal{J}_{B_0} \neq \emptyset$, that is, that the adversary learns at least on bit in each round, then, the output bit is completely determined by the bits that the adversary saw in round $r$, hence, $\mathrm{REAL}_+^r$ and $\mathrm{IDEAL}_+^r$ both determine (and agree on) the output bit $w$. If the adversary does not learn any bit in round $r$ (and, hence, it does not learn any bits in any other round), then the view of the adversary is completely independent of the value of the output bit and since in both worlds this bit is uniform, the probability that it agrees with $\lambda$ is $1/2$ in both cases. We continue our analysis under the added assumption that (besides $\mathrm{REAL}_+^r$ and $\mathrm{IDEAL}_+^r$ being consistent with $\lambda$) the prescribed output bit in both worlds is consistent with $\lambda$. That is, we assume that the bit $w$ that $\mathcal{S}$ obtains from the trusted party $T_{\mathrm{CTWithD}}$ is the same as the output bit selected by Functionality $\mathrm{ShareGenWithAbort}_r$, and that they both agree with $\lambda$.

We next claim that, under the above assumptions (i.e., conditioned on $\mathrm{REAL}_+^r$ and $\mathrm{IDEAL}_+^r$ and the output bit in each model being consistent with $\lambda$), the probability in both models that the adversary sees shares (of honest parties) for the output bit that are according to $\lambda$ is the same.

By the properties of the Shamir secret-sharing scheme, the shares of the honest parties for the bits of sets $J \in \mathcal{J}_{B_0}$ (i.e., such that $|B_0 \cap Q_J| \geq o_J$) are completely determined by the view of the adversary, i.e., by the shares of corrupted parties for these bits (since any $o_J$ shares determine the other shares in a Shamir $o_J$-out-of-$|Q_J|$ scheme). Hence, these shares will be consistent with $\lambda$ with probability 1 in both models. We note that this is true, since the selection of shares is done properly in both model (specifically, in the simulation, the shares for honest parties $X_j^{r,J}$ were already selected in the simulation of round $r$, together with the shares of corrupted parties). For the remaining shares, i.e., shares of honest parties $p_j$ and subsets $J \notin \mathcal{J}_{B_0}$, we argue that in both models they are uniformly distributed over all shares that complete those seen by the adversary to be secret shares of the output bit $w$. This is true since the shares that the adversary sees for this bit do not determine its value and by the properties of secret-sharing schemes. Thus, the distribution over the shares of honest parties $p_j$ and subsets $J \notin \mathcal{J}_{B_0}$ is the same in both models, given the set of shares of corrupted parties for these bits. That is, selecting uniformly over all shares $X_j^{r,J}$ for honest parties that complete the shares of the corrupted parties (that are specified by $\lambda$) to a sharing of $w$ (as is done by the simulator) is equivalent to uniformly selecting shares for all parties to share $w$ and then selecting uniformly over those for which the shares of corrupted parties agree with $\lambda$ (which describes the distribution over shares, implied by the hybrid world random variable).

□

So far we considered the correctness of the simulation process, however, in order to argue that the protocol is secure, we also need to verify that the simulator can be implemented by a probabilistic polynomial time algorithm.

**Claim 6.5.** *The simulator $\mathcal{S}$ described in Section 6.2 is efficient.*

*Proof.* The simulator uses the adversary in a black-box manner and never rewinds it. Furthermore, the adversary is itself a (non-uniform) polynomial time algorithm. In addition, the simulator needs to simulate Functionality $\mathrm{ShareGenWithAbort}_r$ (at most $m - t$ times), Functionality $\mathrm{FairCoinToss}$ (at most once), and Functionality $\mathrm{Reconstruction}$ (at most once). Since all three functionalities are polynomial time computable, their simulation is efficient. Finally, in each round of the simulation (and in the simulation of a normal termination) the simulator needs to select uniformly distributed shares of the bits that the adversary sees. By the properties of Construction 2.8, this can be done efficiently. In addition, in each round of the simulation the simulator needs to sign these shares, which is also done in polynomial time since it holds the signing key. Hence, since the number of rounds in the protocol is polynomial in the security parameter, the whole simulation is done in polynomial time. □

### 6.3.4 Concluding the Proof of Theorem 6

*Proof of Theorem 6.* In Section 6.2 we describe an ideal-world simulator for any hybrid-world adversary for Protocol $\mathrm{CoinToss}_r$. By the combination of Claim 6.2, Claim 6.3, and Claim 6.4, we have that, under the assumption that the adversary never forges a signature of a message, for any pair $(v, c)$, the probability that $\mathrm{REAL} = (v, c)$ is exactly the same as the probability that $\mathrm{IDEAL} = (v, c)$. That is, the global outputs of an ideal-world simulation and of a hybrid-world execution are identically distributed under the above assumption. As explained above, since the probability to successfully forge a signature for any message is negligible, the above two random variables are statistically close even without the assumption that the adversary never successfully forges any message. Finally, by Claim 6.5 the simulator is efficient.

The analysis of the security of Protocol $\mathrm{CoinToss}_r$ is done relative to an hybrid-world with three ideal functionalities. By Canetti [6], the real-world protocol is computationally secure if computationally-secure real-world implementations of these three functionalities exist. This is indeed the case if enhanced trapdoor permutations exist. Hence, the theorem follows. □

# 7 Coin-Tossing Protocol for any Constant Fraction of Corrupted Parties

In this section we describe a coin-tossing protocol for any constant fraction of corrupted parties and we prove the following theorem:

**Theorem 7.** *Let $n$ be the security parameter. If enhanced trapdoor permutations exist, then for any $m = m(n)$, $t = t(n)$, and $r = r(n)$ such that $t = (1 - \varepsilon)m$, for some constant $0 < \varepsilon \leq 1/2$, there is an $r$-round $m$-party coin-tossing protocol tolerating up to $t$ corrupted parties that has bias $O\left(1/\left(\varepsilon\sqrt{r - t}\right)\right)$.*

Before our work, the best known protocol for this scenario is an extension of Blum's two-party coin-tossing protocol [5] to an $r$-round $m$-party protocol that has bias $O\left(t/\sqrt{r - t}\right)$ [2, 8]. In this protocol, in each round $i$ of the protocol, the parties jointly select a random bit $\sigma_i$ in two phases. In the first phase, each party commits to a "private" random bit, and in the second phase the private bits are all revealed and the output bit $\sigma_i$ is taken to be the XOR of all private bits. The output of the whole protocol is the value of

the majority of the $\sigma_i$'s. When there is an abort in round $i$, the remaining parties repeat the computation of round $i$ and continue with the prescribed computation.

Intuitively, the best strategy for a rushing adversary to bias the output of the protocol, say toward 0, is in each round $i$ to instruct a corrupted party to abort before the completion of the revealing phase if $\sigma_i = 1$. This is possible since the rushing adversary learns $\sigma_i$ before the completion of round $i$, specifically, a corrupted party can delay its message until all honest parties reveal their bit. This can go on at most $t$ times, resulting in a total bias of $O\left(t/\left(\sqrt{r}-t\right)\right) = O\left(t/\sqrt{r}\right)$, whenever $r = \Omega(t^2)$.

In our protocol, we follow the general structure of the above protocol in computing the $\sigma_i$'s and taking the majority over them. However, we compute each $\sigma_i$ using a secure-with-abort and cheat-detection protocol, such that either the computation is completed or at least a constant fraction of the corrupted parties abort (specifically, $m - t$ corrupted parties abort).

Next, we briefly describe the computation of each $\sigma_i$ in our protocol. We show how to obtain a constant round secure-with-abort and cheat-detection protocol to compute a random bit that identifies at least $m - t$ cheating parties. Let $m_i$ be the number of active parties at the beginning of round $i$ and $t_i$ be an upper bound on the number of active corrupted parties at the beginning of round $i$ (that is, if $t'$ parties have aborted in rounds $1, \ldots, i-1$, then $t_i = t - t'$). We use a constant round secure-with-abort and cheat-detection protocol. Such a protocol exists under reasonable cryptographic assumptions. More details appear in Appendix A.

In the first phase, a preprocessing phase, the active parties execute a constant round secure-with-abort and cheat-detection protocol to compute a $(t_i + 1)$-out-of-$m_i$ secret-sharing of a random bit $\sigma_i$. That is, at the end of this phase, each party holds a share in a $(t_i+1)$-out-of-$m_i$ Shamir secret-sharing scheme of $\sigma_i$. To confine adversarial strategies to aborts, the share that each party receives is signed and a verification key is given to all parties. In a second phase, a revealing phase, all parties reveal their shares and reconstruct $\sigma_i$. In the reveal phase, broadcasting anything other than a signed share is treated as abort. The formal description of the protocol appears in Figure 7.

To see that the above protocol achieves the required properties, observe that after the first phase the adversary cannot reconstruct $\sigma_i$. Thus, by aborting the preprocessing round, corrupted parties cannot bias the output. We stress that they are able to cause the preprocessing phase to fail, at the cost of at least one corrupted party being detected by all honest parties. In such a case, the preprocessing stage is repeated without the detected party. This, however, can only happen at most $t$ times in total, throughout the whole protocol.

In the revealing phase, a rushing adversary can learn $\sigma_i$ before the corrupted parties broadcast their messages and, thus, can bias the output by not broadcasting these messages. However, by the properties of the secret-sharing scheme, at least $m_i - t_i$ parties have to not broadcast their message, and, hence, effectively abort the computation. Hence, the adversary cannot do this too many times throughout the protocol. The next claim bounds the maximal number of bits the adversary could learn during the protocol.

**Claim 7.1.** *The adversary can prevent the honest parties of learning the bit of $2(1-\varepsilon)/\varepsilon$ rounds throughout the protocol, before an honest majority among active parties is guaranteed.*

*Proof.* In each round $i$, a $(t_i + 1)$-out-of-$m_i$ Shamir secret-sharing scheme is used for sharing $\sigma_i$, therefore, $m_i - t_i$ parties have to abort in order to prevent reviling $\sigma_i$. However, recall that $m_i - t_i$ is a lower bound on the number of honest parties, therefore, $m - t = m_1 - t_1 = m_2 - t_2 = \ldots$. Hence, the number of parties that have to abort in order to prevent the honest parties from reviling the bit is constant times $m$ and equals to the number of honest parties. Assume that the adversary prevented the honest parties of learning the bits of $x$ rounds and after it an honest majority is achieved. Therefore, after an abort of at least $x(m - t)$ corrupted parties, it holds that $t_x \leq t - x(m - t)$ and $m_x \leq m - x(m - t)$. As an honest majority is achieved, i.e.,

**An $m$-party protocol** $\mathrm{CoinTossForConstantFraction}_r$

**Notations:** $1^n$ – the security parameter, $r = r(n)$ – the number of rounds in the protocol, $m = m(n)$ – the number of parties, and $t = t(n)$ – a bound on the number of corrupted parties.

**Joint input:** The security parameter $1^n$.

**Initialization:** $D \leftarrow \emptyset, t_1 \leftarrow t, m_1 \leftarrow m$.

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

1. If $t_i \geq m_i/2$,
   (a) The parties execute a secure-with-abort and cheat-detection protocol computing a functionality in which:
      - Signing and verification keys are produced.
      - A bit $\sigma_i$ is uniformly selected at random.
      - The bit $\sigma_i$ is shared in a $(t_i + 1)$-out-of-$m_i$ Shamir secret-sharing scheme.
      - Each share is signed using the signing key.

      The output of each party is a signed share and the verification key.
      i. If some parties abort, that is, the output of the honest parties is $\{\text{“}\texttt{abort}_j\text{”}\}_{j \in J}$ for some non-empty set $J$, then $t_i \leftarrow t_i - |J|$, $m_i \leftarrow m_i - |J|$, $D \leftarrow D \cup J$, and Step (1) is repeated without the parties whose indices are in $D$.
      ii. Else (no party aborts), set $t_{i+1} \leftarrow t_i$ and $m_{i+1} \leftarrow m_i$.
   (b) The parties reconstruct the bit for this round $\sigma_i$ that was secret-shared in Step (1a):
      i. Each active party broadcasts its signed share.
      ii. If $t_i + 1$ valid shares were sent, each party reconstructs $\sigma_i$.
      iii. Else, a set of aborted parties indexed by $J$ is identified and
         A. Set $t_i \leftarrow t_i - |J|$, $m_i \leftarrow m_i - |J|$, and $D \leftarrow D \cup J$.
         B. Step (1) is repeated without the parties whose indices are in $D$.
2. Else ($t_i < m_i/2$), the active parties execute a secure multiparty protocol with an honest majority in which the bits $\sigma_i, \ldots, \sigma_r$ are selected uniformly at random and each active party receives them. Next, the active parties skip to the end of round $r$.

**At the end of round** $r$**:** Each active party calculates the majority of the reconstructed bits, outputs it, and halts.

Figure 7: Protocol $\mathrm{CoinTossForConstantFraction}_r$.

$t_x < m_x/2$, then the following inequity holds:

$$t - x(m-t) < \frac{m - x(m-t)}{2}.$$

By multiplying by two and moving sides, we get the following inequality:

$$2t - m < x(m-t).$$

By plugging $t = (1-\varepsilon)m$ into the last inequality we get:

$$2((1-\varepsilon)m) - m < x(m-(1-\varepsilon)m).$$

Therefore, the number of rounds $x$ is less than $2(1-\varepsilon)/\varepsilon$. □

Thus, the majority function is applied to $\Omega(r-t)$ random bits, of which the adversary can bias $2(1-\varepsilon)/\varepsilon$. Thus, the total bias of the protocol is $O\left(\frac{1}{\varepsilon\sqrt{r-t}}\right)$.

**Acknowledgments.** We are grateful to Yehuda Lindell for many helpful discussions and great advice. We thank Oded Goldreich and Gil Segev for suggesting this problem and for useful conversations.

# References

[1] Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *J. of Cryptology*, 23(2):281–343, 2010.

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

[3] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proc. of the 22nd ACM Symp. on the Theory of Computing*, pages 503–513, 1990.

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

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

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

[7] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *Proc. of the 34th ACM Symp. on the Theory of Computing*, pages 494–503, 2002.

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

[9] R. Cleve and R. Impagliazzo. Martingales, collective coin flipping and discrete control processes. Online version: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.1797, 1993.

[10] O. Goldreich. *Foundations of Cryptography, Voume I Basic Tools*. Cambridge University Press, 2001.

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

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

[13] S. Goldwasser and Y. Lindell. Secure multi-party computation without agreement. *J. of Cryptology*, 18(3):247–287, 2005.

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

[15] S. D. Gordon and J. Katz. Rational secret sharing, revisited. In *SCN – Security in Communication Networks 2006*, pages 229–241, 2006.

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

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

[18] Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In *Advances in Cryptology – CRYPTO 2006*, number 4117 in Lecture Notes in Computer Science, pages 483–500. Springer-Verlag, 2006.

[19] Y. Ishai, R. Ostrovsky, and H. Seyalioglu. Identifying cheaters without an honest majority. In *Proc. of the Ninth Theory of Cryptography Conference – TCC 2012*, pages 21–38, 2012.

[20] J. Katz. On achieving the "best of both worlds" in secure multiparty computation. In *Proc. of the 39th ACM Symp. on the Theory of Computing*, pages 11–20, 2007.

[21] Y. Lindell. Parallel coin-tossing and constant-round secure two-party computation. *J. of Cryptology*, 16(3):143–184, 2003.

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

[23] R. Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proc. of the 36th ACM Symp. on the Theory of Computing*, pages 232–241, 2004.

[24] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proc. of the 21st ACM Symp. on the Theory of Computing*, pages 73–85, 1989.

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

# A On Achieving Cheat Detection

As discussed in Section 2.2, a key requirement in all our constructions is the existence of constant-round secure with abort protocols with the ability to detect a cheating party. That is, the protocol we use does not only guarantee security with abort, but also that in the case of an abort (or any cheating for that matter), honest parties will be able to detect at least one cheating party. Standard definitions of secure multiparty computation with abort do not guarantee the ability to detect cheating parties. That is, they require that honest parties are notified by the protocol upon an abort, however, there is no concept of pointing out a cheating party. Such a definition of cheat detection was given by Aumann and Lindell [1] in their work on the covert adversary model. Roughly speaking, their definition requires that in the ideal world upon abort the trusted party gives all honest parties an ID of one of the corrupted parties (as chosen by the adversary). The formal definition is given in Section 2.2.

A generic way of obtaining secure protocols is taking a protocol that is secure against a semi-honest adversary and compiling it into a protocol that is secure in the malicious model. Goldreich, Micali, and Wigderson [12] offer a specific paradigm for achieving such compilation. Very roughly, their technique is based on each party proving to the other parties in zero-knowledge that every step it makes is as prescribed in the protocol. In our setting, any protocol obtained by this compilation will have the ability to detect cheating if the zero-knowledge sub-protocols it uses have two properties:

*Perfect completeness.* The verifier has no way to incriminate an honest prover.

*Public verifiability.* Any party (not only the verifier) can verify the correctness of a proof, given the transcript of messages.

Recall that we assume that all communication is sent on an authenticated broadcast channel. Thus, any party that cheats must fail to prove its correctness using a zero-knowledge proof with all but negligible probability. By the above public verifiability property, all other parties can verify that the proof is incorrect, and, hence, detect this party's cheating. By the perfect correctness, a malicious verifier cannot incriminate the prover.

For our needs, it is essential that the protocol has *constant number of rounds*. One can obtain a multiparty constant-round secure-with-abort protocol with the ability to detect cheating by using methods from Pass [23]. Pass presents a construction of bounded concurrent simulation-sound zero-knowledge protocols (that is, the soundness of each one of the protocols is preserved even when all the other protocols are simulated at the same time with the role of the prover and verifier reversed). He shows how these protocols can be used to realize the ideal one-to-many zero-knowledge proof of knowledge functionality, $\mathrm{IDEAL}_{ZK}^{1:M}$ (that is, a functionality where a single prover should prove to all other parties the correctness of some statement). Realizing this functionality, Pass applies the compilation technique of Canetti, Lindell, Ostrovsky, and Sahai [7] (which, in turn, follows the paradigm of [12]) to the constant-round semi-honest protocol of [3]. Thus, Pass's results [23] imply a constant-round multiparty secure-with-abort computation for any feasible functionality. However, this protocol does not immediately imply the ability to detect cheating.

Pass [23] considers a setting where the parties are connected through a point-to-point asynchronous public network. To realize the $\mathrm{IDEAL}_{ZK}^{1:M}$ functionality, he reduces the usage of the $\mathrm{IDEAL}_{ZK}^{1:M}$ functionality to the usage of the more standard two-party ideal zero-knowledge proof of knowledge functionality, $\mathrm{IDEAL}_{ZK}$ (i.e., a functionality that models interaction between a single prover and single verifier). The idea underlying this reduction is similar to the idea used by Goldwasser and Lindell [13] to implement a broadcast channel in a similar setting. However, as is the case with the protocol of [13], this step does not yield the ability to detect cheating since a corrupted party can cause the protocol to terminate without

revealing its identity. Fortunately, we can overcome this problem by using a broadcast channel as well as a synchronous network, thus, simplifying the reduction to $\mathrm{IDEAL}_{ZK}$ to requiring each party to prove in zero-knowledge the same statement to each of the other parties (while other parties monitor the interaction). Furthermore, to verify that the resulting protocol has the ability to detect cheating, it suffices to observe that the zero-knowledge protocols of Pass satisfy the two above-mentioned properties of perfect completeness and public verifiability.[5]

[5]Canetti et al. [6] use common random string. As we use a broadcast channel and we do not require universally composable security, we do not need to use such a string.