Identifying Cheaters Without an Honest Majority

Yuval Ishai [⋆1], Rafail Ostrovsky [⋆⋆2], and Hakan Seyalioglu [⋆⋆⋆3]

[1] Department of Computer Science, Technion
[2] Department of Computer Science and Department of Mathematics, UCLA
[3] Department of Mathematics, UCLA

Abstract. Motivated by problems in secure multiparty computation (MPC), we study a natural extension of *identifiable secret sharing* to the case where an arbitrary number of players may be corrupted. An identifiable secret sharing scheme is a secret sharing scheme in which the reconstruction algorithm, after receiving shares from all players, either outputs the correct secret or publicly identifies the set of all cheaters (players who modified their original shares) with overwhelming success probability. This property is impossible to achieve without an honest majority. Instead, we settle for having the reconstruction algorithm inform each *honest* player of the correct set of cheaters. We show that this new notion of secret sharing can be *unconditionally* realized in the presence of arbitrarily many corrupted players. We demonstrate the usefulness of this primitive by presenting several applications to MPC without an honest majority.

- Complete primitives for MPC. We present the first unconditional construction of a complete primitive for fully secure function evaluation whose complexity does not grow with the complexity of the function being evaluated. This can be used for realizing fully secure MPC using *small* and *stateless* tamper-proof hardware. A previous completeness result of Gordon et al. (TCC 2010) required the use of cryptographic signatures.
- Applications to partial fairness. We eliminate the use of cryptography from the online phase of recent protocols for multiparty coin-flipping and MPC with partial fairness (Beimel et al., Crypto 2010 and Crypto 2011). This is a corollary of a more general technique for unconditionally upgrading security against fail-stop adversaries with preprocessing to security against malicious adversaries.

⋆ Work done in part while visiting UCLA. Supported by ERC Starting Grant 259426, ISF grant 1361/10, and BSF grant 2008411.

⋆⋆ Research supported in part by NSF grants 0830803, 09165174, 1065276, 1118126 and 1136174, US-Israel BSF grant 2008411, B. John Garrick Foundation, OKAWA Foundation, IBM, Lockheed-Martin Corporation and the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

⋆⋆⋆ Research supported by a NSF Graduate Research Fellowship.

Finally, we complement our positive results by a negative result on identifying cheaters in unconditionally secure MPC. It is known that MPC without an honest majority can be realized *unconditionally* in the OT-hybrid model, provided that one settles for "security with abort" (Kilian, 1988). That is, the adversary can decide whether to abort the protocol after learning the outputs of corrupted players. We show that such protocols *cannot* be strengthened so that all honest players agree on the identity of a corrupted player in the event that the protocol aborts, even if a broadcast primitive can be used. This is contrasted with the computational setting, in which this stronger notion of security can be realized under standard cryptographic assumptions (Goldreich et al., 1987).

1 Introduction

Consider a scenario in which n mutually distrustful clients wish to distribute a long computation. Instead of directly interacting with each other, they rely on a trusted external *stateless* server. In each invocation, the server receives a share of the current state of computation (and possibly an additional input) from each client, and returns a share of the new state (and possibly an output) to each client. This scenario may apply to distributing sensitive computations using servers in the cloud, where requiring servers to maintain state information between different invocations is undesirable for security reasons.

The question we ask is what form of secret sharing is suitable for distributing the joint state between the clients. Naturally, we do not want to assume that a majority of the clients are honest (this rules out fair [8] or unconditionally secure [6] solutions that use direct interaction between the clients and do not employ the server). Additively sharing the state fails in protecting the correctness of the computation, allowing each client to change the global state without being detected. A better solution is to use *robust* secret sharing that can *detect* cheating (cf. [29, 9] and references therein). When there are three or more clients, this too has the disadvantage that it offers no strong deterrent against cheating: while cheating does not go undetected, it disrupts the computation without identifying a corrupted client. This motivates the use of *identifiable secret sharing*, where a failure of the reconstruction algorithm results in identifying the clients who modified their shares.

Identifiable secret sharing as above can be realized when a majority of the clients are honest [22, 20, 7, 24]. But without an honest majority, there is no way for the server to tell apart cheaters from honest clients. Indeed, $n/2$ cheaters can simulate a consistent sharing of an incorrect secret, which makes it impossible for the server to tell which of the two sets of consistent shares is correct. However, this does not rule out the alternative of allowing the server to *inform each client* (with negligible error probability) which shares have been modified *assuming that this client is honest*. We refer to this as *locally-identifiable secret sharing* (LISS). Note that except with negligible probability, each honest client will agree on which clients are corrupted and should be disqualified. Thus use of LISS to share the state minimizes the incentive to cheat and allows the honest clients

in the event of reconstruction failure to *agree* on a strict subset of the clients that includes all honest clients. This subset has the option of restarting the computation on their original inputs, using default values for the inputs of the remaining clients, without losing in this process any of the honest clients.

Settling for *computational* security, LISS can be realized via the use of *digital signatures*: the sharing procedure distributes to all clients *the same* public verification key vk, and gives to each client a signature of its additive share of the secret using the corresponding secret key sk. Reconstruction proceeds by letting each client send to the server its original share, vk, and the signature on the share. The server can then identify definite cheaters as those who supply an inconsistent triplet, and partition the remaining clients according to the value of vk they provide. In fact, such a computationally secure LISS scheme was implicitly used by Gordon et al. [13] in the context of defining a complete primitive for MPC. The possibility of an unconditionally secure construction remained open. This question is motivated not only by the goals of enhancing security and eliminating assumptions, but also by the potential *efficiency* advantages of information-theoretic techniques. This is especially significant in applications (such as those discussed below) where the share generation process is distributed between multiple players.

1.1 Our Results

Constructions. Our main result is an affirmative answer to the above question: we present an unconditional construction of an n-out-of-n LISS scheme whose security holds in the presence of an arbitrary number of corrupted players. More generally, we show how to efficiently transform any secret sharing scheme into one in which the reconstruction function reveals to every honest player of the identity of *all* shares that have been tampered with. In particular, all honest players agree on the same set of cheaters.

We also consider a weaker variant of LISS that we call *unanimously identifiable secret sharing* (UISS) in which only the latter agreement property is required. That is, if reconstruction fails, all honest players should agree on the same (non-empty) set of cheaters. This weaker primitive is easier to construct. (In fact, a construction of UISS is implicit in [25].) In contrast to LISS, however, UISS does not guarantee that *all* cheaters are detected in the event that reconstruction fails.

Applications. We present several applications of the above primitives in the context of MPC without an honest majority. In the following, the term MPC refers to the special case of secure function evaluation, namely MPC of non-reactive (stateless) functionalities. We use poly and neg to represent polynomial and negligible functions, respectively, and κ denote a statistical security parameter. While we mainly consider statistical security, our results are also useful in the domain of computational security.

Complete primitives for MPC. It is well known that fully secure MPC (with fairness and guaranteed output delivery) is impossible to achieve in general

without an honest majority [8]. This naturally raises the question of finding a minimal *complete* primitive that can be used to get around this limitation. Such a primitive is defined by a (stateless) deterministic functionality g mapping n inputs to n outputs, such that any n-party functionality f can be realized using a trusted instance of g initialized between every tuple of players that can supply input to it. The first such results characterized complete boolean primitives for MPC with security against a *passive* adversary [21, 19]. In the case of active adversaries, Fitzi et al. [11] presented a complete primitive for fully secure MPC whose computational complexity grows linearly with complexity of f. This left open the question of finding a "simple" complete primitive, whose complexity does not depend on the complexity of f. One such primitive was given by Gordon et al. [13] using digital signatures. We use UISS to get an unconditional variant of this result. In this variant, the complexity of g only grows with the *output length* of f.

Theorem 1. *There is a deterministic, polynomial-time computable functionality g with input and output size $\mathsf{poly}(n, \kappa, \beta)$ such that any n-party function f computed by a circuit of size σ and output length β can be realized with full statistical security (and $2^{-\kappa}$ simulation error) using $\mathsf{poly}(n, \sigma)$ calls to g.*

This result has an interesting interpretation in the context of a recent line of work on basing cryptography on tamper-proof hardware (see [17, 15] and references therein). In this line of work, several impossibility results in cryptography (including UC security, unconditional security, software protection and obfuscation) were circumvented by using tamper-proof hardware tokens. These works spent efforts on minimizing the *size* of the tokens, employing *stateless* (rather than stateful) tokens, and minimizing or eliminating cryptographic assumptions. The above result can be viewed as achieving all these goals simultaneously in the context of another major impossibility result: the impossibility of fully secure MPC without an honest majority. It implies that a *small* and *stateless* token, connected via secure channels to the n players, suffices to *unconditionally* realize fully secure MPC. We note that connecting the same token to *all* players is necessary, as implied by the results of Fitzi et al. [11].

We also present other variants of the previous completeness theorem which rely on computational assumptions but still avoid the use of cryptography inside the primitive. These variants have the advantage of requiring only a small number of calls to the primitive (independently of the complexity of f).

Applications to partial fairness. A recent line of works studies the extent to which *partial fairness* can be achieved in MPC without an honest majority. Partial fairness can be defined by restricting the simulation error to be small (e.g., inverse polynomial) but not negligible [14]. We show that in partially fair protocols of Beimel et al.[2, 1] (extending previous two-party protocols of Moran et al. [23] and Gordon and Katz [14]), the use of a digital signature scheme can be replaced by a unanimously identifiable commitment scheme, a second primitive we define that can be used as a substitute for LISS in certain applications. This yields *unconditional* multiparty protocols for coin-flipping and MPC with partial

fairness in the preprocessing model, namely assuming that players have offline access to correlated randomness. We note that trusted preprocessing does not trivialize the problem, because the output needs to be unpredictable in the end of the preprocessing phase. In fact, the negative results on achieving full fairness apply to the preprocessing model as well. The preprocessing model does allow, however, to eliminate the assumptions of secure channels and broadcast, which can be implemented unconditionally in the preprocessing model [27].

The preprocessing phase can be realized either by a trusted offline dealer or via a distributed protocol (possibly employing additional parties for unconditional security). Even if one relies on a computationally secure protocol for distributing the preprocessing phase, the protocols we get have the advantage of making only a *black-box* use of the underlying cryptographic primitives, whereas the original protocols from [2, 1] make a non-black-box use of a one-way function.

In the case of coin-flipping, applying our primitive to the offline dealer protocol from [2] implies the following:

Theorem 2. *Assume preprocessing by a trusted off-line dealer. Fix constants n and t such that $t < 2n/3$. Then, for any r, there is an r-round n-party unconditionally secure coin-tossing protocol over point-to-point channels tolerating up to t malicious players with bias $O(1/r)$.*

Our results on MPC with partial fairness are obtained via a general technique for unconditionally upgrading security against fail-stop adversaries to security against malicious adversaries where the messages sent by the players are determined in the preprocessing stage.

A negative result. It is known that MPC without an honest majority can be realized *unconditionally* in the OT-hybrid model, provided that one settles for "security with abort" [18, 16]. That is, the adversary can decide whether to abort the protocol after learning the outputs of corrupted players but before the honest players receive their output. We show that such protocols *cannot* be strengthened so that all honest players agree on the identity of a corrupted player in the event that the protocol aborts, even if a broadcast primitive and trusted access to an arbitrary pairwise functionality is assumed. This is contrasted with the computational setting, in which this stronger notion of security can be realized under standard cryptographic assumptions [12]. Our negative result strengthens a previous negative result from [11], which shows that pairwise functionalities alone (without broadcast) are not sufficient in general for *fully secure* n-party computation. For lack of space, the details of this result are deferred to the full version.

2 Preliminaries

Our communication model allows for authenticated point to point and broadcast channels unless specified otherwise. While we define our algorithms in terms of finite sets (with fixed input size) and fixed error rate, they can be implemented by uniform algorithms that are polynomial in the bit-length of the inputs, the

number of players, and the statistical security parameter κ guaranteeing $\delta = 2^{-\kappa}$ error. The latter is the default convention whenever no value of δ is specified.

We only consider non-adaptive adversaries but our secret sharing definitions and proofs can easily be extended to the adaptive case. We denote the n players by $\mathcal{P} = \{P_1, P_2, \dots P_n\}$ and will often identify a player with its index. A collection of subsets $\mathbb{A}$ of $\mathcal{P}$ will be called *monotone* if for any $B \in \mathbb{A}$, if $B \subseteq C \subseteq \mathcal{P}$ then, $C \in \mathbb{A}$. We let $[n]$ denote the set $\{1, \dots, n\}$. We use $x \xleftarrow{\$} X$ to denote a uniform choice of x from a set X.

2.1 Secret Sharing

We briefly describe our notation for standard secret sharing schemes. A secret sharing scheme is defined by a pair of algorithms $(\mathsf{Share}, \mathsf{Rec})$, where Share is a randomized algorithm mapping a secret from S to the share space $\prod_{i=1}^{n} S_i$, and Rec is a deterministic reconstruction algorithm mapping the shares of a qualified set of players (along with the identity of this set) to a secret from S. We will refer to S as the secret space and to S_i as the share space of P_i. An *access structure* is a monotone collection of player sets. We say that a secret sharing scheme realizes an access structure $\mathbb{A}$ if sets in $\mathbb{A}$ can reconstruct the secret s and others can learn nothing about it. Throughout this work we define secret sharing schemes to have perfect *correctness* (authorized sets always correctly reconstruct the secret) and perfect *secrecy* (the shares of unauthorized sets reveal no information about the secret). For all additional security guarantees we assume the adversary knows the secret that is being shared; even if the secret is compromised, the adversary should not be able to cause the reconstruction algorithm to behave undesirably (e.g. by outputting an incorrect secret or implicating an honest player of cheating), except with small probability.

As usual, we consider a single adversary who may corrupt one or more players. We distinguish between passive and active corruptions using the following terminology.

Definition 1. (Tampering) *A corrupted player is said to have* tampered *with its share if it provides to the reconstruction algorithm a share different than the one assigned by the distribution algorithm. Such a share is called a* tampered share *and such a player is called a* cheater.

Identifiable Secret Sharing. An identifiable secret sharing scheme is a secret sharing scheme in which the reconstruction algorithm can identify all cheaters in the event that it fails to reconstruct the secret. The above guarantee should hold except with some failure probability δ as long as there are at most t cheaters for an additional parameter t. In our definition we assume that the tampering is done by a single adversary who can observe the shares of a set C of up to t corrupted players and based on this information decide on how to tamper with their shares.

Definition 2. (Identifiable Secret Sharing) *A secret sharing scheme realizing $\mathbb{A}$ is (δ, t)-identifiable if for any (unbounded) adversary A and any $s \in S$, the success probability of A in the following game is at most δ:*

1. $(s_1, s_2, \ldots, s_n) \leftarrow \mathsf{Share}(s)$;
2. A outputs a set $C \subset [n]$ such that $|C| \leq t$ and receives $(s_j)_{j \in C}$;
3. A outputs $(B, (s'_j)_{j \in C \cap B})$ where $B \in \mathbb{A}$;
4. $\mathsf{Out} \leftarrow \mathsf{Rec}(B, (t_j)_{j \in B})$ where $t_j = s'_j$ if $j \in C$ and $t_j = s_j$ otherwise.

A succeeds if for some $j \in C \cap B$, $s'_j \neq s_j$ and $\mathsf{Out} \neq (\perp, \{P_i \in C \cap B : s'_i \neq s_i\})$.

The first work on identifiable secret sharing is due to McEliece and Sarwate [22] who showed that Shamir's k-threshold secret sharing scheme allows perfect identification if $k + 2t$ players of which at most t are cheaters are involved in reconstruction. Several works consider various relaxations of identifiability [28, 3, 5] which suffice for some applications but are not suitable for MPC with a dishonest majority. There is also substantial work on the efficiency of identifiable secret sharing [20, 24, 7].

Identifiability is not possible with a dishonest majority for a simple reason: If half of the participants are dishonest they can run the sharing algorithm independently among themselves and return as their shares the output of the second run of the algorithm. This strategy makes it impossible for Rec to identify which half of the shares come from the first run of the Share algorithm and which come from the second since they are run independently. This is captured by the following theorem (see full version for proof):

Theorem 3. (No identifiability with a dishonest majority) *For any $t, n, S, \mathbb{A}$ with $t \geq n/2$, $|S| \geq 2$, $\mathbb{A} \neq \emptyset$, there is no $(1/4, t)$-identifiable secret sharing scheme with secret space S and access structure $\mathbb{A}$.*

3 Locally Identifiable Secret Sharing

We now give our relaxation of identifiable secret sharing that can be realized when arbitrarily many players may be corrupted. Informally, the guarantee we require is that if the reconstruction fails, the reconstruction algorithm outputs a tuple of players to each player P_i with the guarantee that if P_i is honest, the tuple returned to P_i is precisely the players that tampered with their shares. While this is equivalent to identifiability from the point of view of the honest players, it allows us to circumvent the impossibility result of Theorem 3. Note that we define LISS as being a special type of secret sharing scheme, so the usual correctness and secrecy requirements should hold in addition to the requirements detailed below.

Definition 3. (Lists) *Throughout this paper when we refer to a list $\mathcal{L}$ we refer to a subset of the players in the protocol ($\mathcal{L} \subset \{P_1, P_2, \ldots, P_n\}$).*

Definition 4. (LISS) *A secret sharing scheme realizing $\mathbb{A}$ is locally δ-identifiable if it satisfies the following requirements:*

- **Unanimity:** *For any adversary A and $s \in S$, the probability of A's success in the following game is at most δ:*
  1. $(s_1, s_2, \ldots, s_n) \leftarrow \mathsf{Share}(s)$;
  2. *A outputs a set $C \subset [n]$ to corrupt and then receives $(s_i : i \in C)$;*
  3. *A outputs $(B, (s'_j)_{j \in C \cap B})$ such that $B \in \mathbb{A}$ and $B \not\subset C$;*
  4. $\mathsf{Out} \leftarrow \mathsf{Rec}(B, (t_j)_{j \in B})$ *where $t_j = s'_j$ if $j \in C$ and $t_j = s_j$ otherwise.*

  The adversary succeeds unless:
  1. *Reconstruction succeeds: $\mathsf{Out} = s$ or,*
  2. *Each honest player's list is the list of all cheaters: $\mathsf{Out} = (\perp, (\mathcal{L}_j)_{j \in B})$ where for all $j \in B \setminus C$, $\mathcal{L}_j = \{P_i \in C \cap B : s'_j \neq s_j\}$.*
- *The scheme has* **Predictable Failures** *(Definition 5).*

We briefly motivate the requirement of Predictable Failures before defining it. The problem to address is that the additional outputs $\mathcal{L}_j$, or even the event of not reconstructing the secret, may leak some information concerning the secret unless a separate guarantee is made. This can cause a problem in applications and therefore we must have a way to simulate the actions of Rec in the case of tampering. Note that this is a new issue not present in identifiable secret sharing: As the Rec function does not simply output a list of tampering players, there are no a-priori guarantees concerning the lists corresponding to dishonest players and therefore we must make requirements on them separately.

Definition 5. (Predictable Failures) *A secret sharing scheme has* δ-Predictable Failures *if there is an algorithm SRec such that for any adversary A and $s \in S$, the probability of success in the following game is less than δ:*

1. $(s_1, s_2, \ldots, s_n) \leftarrow \mathsf{Share}(s)$;
2. A outputs a set $C \subset [n]$ to corrupt and receives $(s_i)_{i \in C}$;
3. A outputs $(B, (s'_j)_{j \in C \cap B})$ such that $B \in \mathbb{A}$ and $B \not\subset C$;
4. $\mathsf{SOut} \leftarrow \mathsf{SRec}(C, B, (s_i)_{i \in C}, (s'_i)_{i \in C \cap B})$;
5. $\mathsf{Out} \leftarrow \mathsf{Rec}(B, (t_j)_{j \in B})$ where $t_j = s'_j$ if $j \in C$ and $t_j = s_j$ otherwise.

A succeeds unless:

1. SRec correctly predicts success: $\mathsf{SOut} = \textsc{Success}$ and $\mathsf{Out} = s$ or,
2. SRec predicts the output of Rec: $\mathsf{SOut} = \mathsf{Out} \neq s$.

3.1 Our Construction

Let $(\mathsf{Sh}, \mathsf{Rc})$ be a secret sharing scheme realizing access structure $\mathbb{A}$ with $\mathsf{Sh} : S \to \mathbb{F}^n$ where $\mathbb{F}$ is a field. Let $I_{n \times n}$ and $0_{n \times n}$ denote the identity and all zero matrix respectively. We use $\mathbb{F}^{n \times n}$ to denote the set of all $n \times n$ matrices with elements in $\mathbb{F}$ and $GL_n(\mathbb{F})$ to denote the set of all such invertible matrices. For a matrix M we will use $M(i, j)$ to denote the (i, j) entry of M. By default we

assume vectors to be column vectors, we will use the notation $\boldsymbol{a}^T$ when referring to a row vector. We use the notation $\mathbb{F}^*$ to denote $\mathbb{F} \setminus \{0\}$.

Share(s):

1. Generate $(t_1, t_2, \ldots, t_n) \leftarrow \mathsf{Sh}(s)$, $u_i, v_i \xleftarrow{\$} \mathbb{F}^*$ for all $i \in [n]$;
2. Define $C_0 \in \mathbb{F}^{n \times n}$ as $\begin{cases} C_0(i,j) = u_i^{j+1} v_j^{i+1} + u_i v_j + 1 & \text{for } i \neq j; \\ C_0(i,i) = t_i & \text{for } i \in [n]. \end{cases}$
3. Define C blockwise as: $\begin{pmatrix} C_0 & I_{n\times n} \\ I_{n\times n} & 0_{n\times n} \end{pmatrix}$;
4. Generate $B \xleftarrow{\$} GL_{2n}(\mathbb{F})$ and define $A = CB^{-1}$;
5. Label row i of A as $\boldsymbol{a}_i^T$ and column j of B as $\boldsymbol{b}_j$;
6. Return $(s_i = (\boldsymbol{a}_i^T, \boldsymbol{b}_i, u_i, v_i))_{i \in [n]}$.

Rec$(D, (s_i = (\boldsymbol{a}_i^T, \boldsymbol{b}_i, u_i, v_i))_{i \in D})$ with $D \in \mathbb{A}$, $\boldsymbol{a}_i^T, \boldsymbol{b}_i \in \mathbb{F}^{2n}, u_i, v_i \in \mathbb{F}^*$:

1. If for all $i \neq j$, $\boldsymbol{a}_i^T \boldsymbol{b}_j = u_i^{j+1} v_j^{i+1} + u_i v_j + 1$:
   - Set $\boldsymbol{a}_i^T \boldsymbol{b}_i = t_i$ for all $i \in B$;
   - Return $\mathsf{Rc}(D, t_i : i \in D)$.
2. Else, for all $i \in D$ set:

$$\mathcal{L}_i = \{P_j : \boldsymbol{a}_i^T \boldsymbol{b}_j \neq u_i^{j+1} v_j^{i+1} + u_i v_j + 1 \text{ or } \boldsymbol{a}_j^T \boldsymbol{b}_i \neq u_j^{i+1} v_i^{j+1} + u_j v_i + 1\};$$

3. Return $(\perp, (\mathcal{L}_i)_{i \in D})$.

Theorem 4. *If $\delta > n^2(n+1)/(|\mathbb{F}| - 1)$, the scheme described above is a δ-LISS scheme realizing $\mathbb{A}$ with secret space S and share space $S_i = \mathbb{F}^{4n+2}$.*

Corollary 1. *Suppose there is a secret sharing scheme which realizes an n-party access structure $\mathbb{A}$ with secret space S and share length β. Then, for any $\delta > 0$ there is a δ-LISS with the same $\mathbb{A}$ and S whose share length is $O(n \log(n/\delta) + n\beta)$.*

Outline of Security. A full proof of security is provided in the full version of this paper but we provide a brief intuition in this section for self containment. We first argue secrecy. Notice that the value t_i is only used in generating the row $\boldsymbol{a}_i^T$, therefore any set of players $E \notin \mathbb{A}$ will have its shares generated using only the t_i values such that $P_i \in E$. The fact that the joint distribution of these t_i values do not depend on the underlying secret (due to the perfect secrecy of $(\mathsf{Sh}, \mathsf{Rc})$) implies secrecy.

Consider now an adversary that is attempting to tamper the share of some P_i (and possibly others) - we will argue that any such attempt will cause the check in Rec between P_i and any honest player P_j to fail with high probability. Assume that the adversary is tampering $\boldsymbol{b}_i \to \boldsymbol{b}_i'$, $v_i \to v_i'$ with one of these values changed (a similar argument will hold if the adversary is tampering $\boldsymbol{a}_i^T$

or u_i). There are then two cases, either $\boldsymbol{b}'_i$ is in $\mathsf{Span}(\{\boldsymbol{b}_i\}_{i\in T})$ where T is the set of corrupted players or it is linearly independent of these values. If $\boldsymbol{b}'_i$ is linearly independent it can be shown that $\boldsymbol{a}_j^T\boldsymbol{b}'_i$ is essentially uniformly distributed over $\mathbb{F}$ conditioned on the view of the adversary even after u_j is fixed and therefore the probability that reconstruction succeeds will be very low (showing this statement is non-trivial).

On the other hand, consider the case where $\boldsymbol{b}'_i \in \mathsf{Span}(\{\boldsymbol{b}_k\}_{k\in T})$. Let $\boldsymbol{b}'_i = \sum_{k\in T}\beta_k\boldsymbol{b}_k$ where $\beta_k \in \mathbb{F}$. Now, the check will succeed only if:

$$\boldsymbol{a}_j^T\sum_{k\in T}\beta_k\boldsymbol{b}_k = u_j^{i+1}v'^{j+1}_i + u_jv'_i + 1 \Leftrightarrow$$

$$\sum_{k\in T}\beta_k(u_j^{k+1}v_k^{j+1} + u_jv_k + 1) = u_j^{i+1}v'^{j+1}_i + u_jv'_i + 1.$$

Similar to our argument of secrecy, the value u_j is uniformly distributed conditioned on every view of the adversary. Therefore, the check in Rec will succeed only if the above equality is satisfied by a uniformly chosen $u_j \in \mathbb{F}^*$. This will happen rarely unless the polynomials on the left and right of the equality (considered as a polynomial in u_j) are equal. For this equality to hold, we must have $\beta_k = 0$ for all $k \neq i$ since otherwise $v_k \neq 0$ would make the polynomials different. Next notice that we must have $\beta_i = 1$ for the constant terms to match. Finally, the u_jv_k term on the left implies that $v'_i = v_i$. Therefore, unless $v'_i = v_i$ and $\boldsymbol{b}'_i = \boldsymbol{b}_i$ this equality will only occur with low probability, which implies that if P_i tampers with either the v_i or $\boldsymbol{b}_i$ value, it will be detected and placed on P_j's list with high probability for all honest P_j. A similar argument holds if either the $\boldsymbol{a}_i^T$ or u_i value is tampered since the method of generation is equivalent to first generating $A \in GL_{2n}(\mathbb{F})$ and setting $B = A^{-1}C$ since C is always invertible.

Notice that we have actually argued that a dishonest player who modifies his share will be on the list of every honest player and symmetrically that all honest players will be on the list of such a dishonest player with high probability. This implies Predictable Failures since an adversary can easily tell which dishonest players will be on a given dishonest player's list from the shares it has, as well as whether or not the honest players will be on his list depending on whether or not the player modifier his share.

4 Relaxing Local Identifiability

In this section we define a new commitment primitive, *unanimously identifiable commitments* that can be used as a leaner substitute for LISS in certain applications. Additionally, we note that this commitment primitive implies a weaker variant of LISS (called *unanimously identificable secret sharing*) that can also be used in our applications to MPC.

4.1 Unanimously Identifiable Commitments

A *unanimously identifiable commitment* (UIC) scheme has a single player (called the sender) committed to a value $s \in S$ by having a trusted dealer send commitments c_i to all other players in the protocol and decommitment information d to the sender such that any tampering of the d value will cause all honest players to either reconstruct the original secret or fail reconstruction simultaneously. As with standard commitments, $(c_i)_{i\in[n]}$ should leak no information concerning s.

Definition 6. (Unanimously Identifiable Commitments) *A δ-UIC scheme consists of a randomized algorithm* Offline *and a deterministic algorithm* Decommit *with the following syntax:*

1. **Offline***: $S \to C^n \times D$. Takes as input a secret $s \in S$ outputs n commitments $c_1, c_2, \ldots, c_n$ and decommitment information d.*
2. **Decommit***: $C \times D \to S \cup \{\perp\}$. Takes as input c_i and the decommitment information d and recreates the secret s or outputs $\perp$ indicating failure.*

Where the algorithms (Offline, Decommit) should satisfy:

- **Completeness.** For any $s \in S$, if $\Pr[\mathsf{Offline}(s) = (c_1, c_2, \ldots, c_n, d)] > 0$ then, $\mathsf{Decommit}(c_i, d) = s$ for any $i \in [n]$.
- **Secrecy.** The values $c_1, c_2, \ldots, c_n$ reveal no information concerning s. Formally, for any $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ and any $s, s' \in S$, the probability that the first n values of $\mathsf{Offline}(s)$ is $\boldsymbol{c}$ is equal to the probability that the first n values of $\mathsf{Offline}(s')$ is $\boldsymbol{c}$.

We now present the final requirement placed on this primitive for use in our applications. In the full version of this paper, we include further intuition to the necessity of this condition but omit it here for space restrictions.

There exists simulators W_1, W_2 such that the two guarantees described below hold with probability at least $1-\delta$ for any A. Consider the following experiment:

1. The adversary, A outputs a set $T \subset [n] \cup \{Q\}$ of players to corrupt;
2. $(c_1, c_2, \ldots, c_n, d) \leftarrow \mathsf{Offline}(s)$;
3. For all $i \in T \cap [n]$ send c_i to the adversary, if $Q \in T$ send d to the adversary;
4. If $Q \notin T$, set $dec = d$; otherwise, dec is output by A.
5. For all $i \in T \cap [n]$, A outputs (c_i', i), fake commitment information for P_i.

The guarantees around this experiment are as follows:

- **Binding with Agreement on Abort.** $\mathsf{Decommit}(c_i, dec) = s$ for all P_i uncorrupted or $\mathsf{Decommit}(c_i, dec) = \perp$ for all P_i uncorrupted.
- **Simulatable Abort.** Let V be the view of A at the end of 5., then:
  1. If A corrupted Q:
     $W_1(V)$ correctly predicts if $\mathsf{Decommit}(c_i, dec) = \perp$ for all $i \in [n]$.
  2. Otherwise:
     $W_2(V, c_i')$ correctly predicts if $\mathsf{Decommit}(c_i', d) = \perp$ for each $i \in T \cap [n]$.

4.2 A Unanimously Identifiable Commitment Scheme

Let $\mathbb{F}$ be a field. We now give a simple construction of a δ-UIC scheme with $S = \mathbb{F}$, $C = \mathbb{F}^{n+2}$ and $D = \mathbb{F}^2$.

Offline(s) :

1. Generate $P(X)$, a random $n+1$ degree polynomial over $\mathbb{F}$ such that $P(0) = s$;
2. For all $i \in [n]$ generate $x_i \xleftarrow{\$} \mathbb{F}$ and let $y_i = P(x_i)$;
3. Set $c_i = (x_i, y_i)$ and $d = P(X)$. Return $((c_i)_{i\in[n]}, d)$.

Decommit$(c_i = (x_i, y_i), d = P(X)$ of degree $n + 1)$: If $P(x_i) \neq y_i$ return $\perp$. Else, return $P(0)$.

Theorem 5. *Let $|\mathbb{F}| > (n+1)^2\delta^{-1} + 1$. The scheme described above is a δ-UIC with $S = \mathbb{F}$, $C = \mathbb{F}^{n+2}$ and $D = \mathbb{F}^2$.*

Related Concepts. In our applications, we mainly use UIC as a substitute for digital signatures. There are some other unconditional notions that have also been introduced for similar purposes (such pseudosignatures [26], distributed commitments [10] and IC signatures [25]). While the construction itself is not novel (for example, it is used in [25]), the property that all of the honest players accept or reject the same commitment is crucial to our applications and differs from the guarantees placed on the other primitives.

4.3 Unanimously Identifiable Secret Sharing

We note that *unanimously identifiable commitments* actually imply a weaker notion of LISS which we call *unanimously Identifiable Secret Sharing* (UISS). The security requirements for a UISS scheme are identical to the requirements to LISS except that the requirement:

• *Each honest player's list is the list of all cheaters:*

is replaced by the requirement:

• *Each honest player's list is the same subset of corrupted players*:

$$\mathsf{Out} = (\mathcal{L}_j)_{j\in B} \text{ where for all } j, j' \in B \setminus T,\ \mathcal{L}_j \subset T \text{ and } \mathcal{L}_j = \mathcal{L}_{j'}.$$

All other requirements remain unchanged, including the requirement of predictable failures. Implementing UISS for access structure $\mathbb{A}$ using a UIC scheme is straightforward by having each user commit to its share. Note that for most applications, UISS can take the role of LISS, at the cost of not necessarily identifying *all* tampered shares if reconstruction fails.

5 Applications

A secure multiparty computation (MPC) protocol allows a set of players to compute a function evaluated on their individual inputs while revealing no information other than the output of the function. We assume familiarity with (standalone) MPC throughout this section and refer the reader to [4] for formal definitions.

5.1 Model of Computation

By default, we consider static, computationally unbounded adversaries who may corrupt up to t of the n parties ($t = n$ by default). We consider both *active* adversaries, who may arbitrarily control the corrupted players, *passive* adversaries, who can only observe the internal state of corrupted players, and *fail stop* adversaries who behave like passive adversaries except that they make corrupted players stop sending messages. Our network model is synchronous with point-to-point channels and a broadcast channel.

The security of an MPC protocol with respect to an ideal functionality f is defined by comparing a real world execution of the protocol to an *ideal model* execution where a trusted party evaluates f. By default, we refer to *statistical* security, where the statistical advantage of distinguishing between the real world and the ideal model execution is bounded by $2^{-\kappa}$ for a statistical security parameter κ. We will only consider the case of secure function evaluation, in which f is stateless. We will mostly consider *fully secure* MPC in which the ideal model adversary cannot prevent the trusted party from sending the outputs of f to the honest players. Full security cannot be achieved even for simple functionalities such as coin-flipping [8] without an honest majority or other assumptions we will discuss. This impossibility holds even with trusted preprocessing; however, in the latter model the assumptions of secure point-to-point channels and a broadcast primitive are unnecessary as they can be implemented unconditionally [27].

5.2 Complete Primitives for MPC

An n-party functionality g is called a *complete primitive* for n-party MPC if it is possible to securely realize any n-party functionality f in the *g-hybrid model,* namely by using ideal calls to g. Here we consider security against an active adversary who may corrupt an arbitrary number of players.

In prior works, such primitives either depend on the complexity of the function being evaluated [11] or rely on cryptographic assumptions [13]. It remained open to construct an *unconditionally* complete primitive whose complexity is independent of the complexity of the evaluated function f. In the following section, we show how to construct such a primitive. Our contribution can be seen as identifying a cryptographic LISS scheme implicitly present in the construction of Gordon et al. [13] and replacing it with an unconditional construction. In fact, it suffices for this purpose to rely on UISS rather than LISS. For simplicity, we

assume that the functionality f being evaluated using g delivers the same output to all players; the general case is handled similarly.

Unconditional Primitive. The first primitive we present is complete for statistically secure MPC and its complexity depends only on the output length of the evaluated functionality f. We give an informal description of the primitive in this version and defer further details to the full version. For expository purposes, we will describe three separate primitives that make up the three modes of operation for the complete primitive.

- FCR_1^1 - Takes as input a bit from a player, runs an n-out-of-n UISS sharing algorithm on this bit and distributes the shares amongst all players.
- FCR_2^1 - Takes as input two n-tuples of shares from the UISS scheme. Internally, the primitive reconstructs the underlying secrets of each, evaluates the NAND of the two secrets, and re-shares the output using the UISS scheme. If reconstruction fails, the functionality will use the lists $\mathcal{L}_j$ output by the UISS scheme to partition the players: If any player is on his own list, the functionality declares this player is disqualified and his input is replaced by a default value by all players. If not, the functionality outputs a partition of the players: P_i and P_j remain in the same partition if $\mathcal{L}_j = \mathcal{L}_i$.
- FCR_3^1 - Takes as input β separate n-tuples of UISS shares, where β is an output length parameter. The functionality either reconstructs each secret and broadcasts all the reconstructed bits, or, if some reconstruction fails, partitions the players as in the previous mode using the first instance of failed reconstruction.

Note that while the first two primitives are randomized, they can be made deterministic by using a standard reduction: the internal randomness can be securely emulated by taking the XOR of shares contributed by the n players.

Using the above primitive, one can securely evaluate any boolean circuit C, which consists of NAND gates and has β output bits, in the following way. The players first use FCR_1^1 to share each of their input bits. After this phase is completed, for each gate in C the players use FCR_2^1 to evaluate shares of the value of each internal value in C. Finally, the players feed the shares of the output values to FCR_3^1 and receive the outputs of C.

Notice that any deviation from the above protocol will result in all honest players identifying the same set of cheaters, and therefore their lists $\mathcal{L}_i$ will be identical. In this case, they are partitioned and the protocol is re-started with default values substituted for the inputs of the corrupted players. Due to the guarantees of the UISS scheme, the partitions can be simulated given only the views of the corrupted players. Defining the three modes of operation as one primitive that can be called on only some partition of the players requires some additional technical steps to fit in with our model of one trusted primitive. In addition to the players declaring which mode they are using, the primitive should also take as input from each player the set of players this player still trusts (as in [13]), we detail this in the full version of this paper.

The above complete primitive yields the following theorem.

Theorem 6. *There is a deterministic, polynomial-time computable functionality g with input and output size $poly(n, \kappa, \beta)$ such that any functionality f computed by a circuit of size σ and output length β can be realized with full statistical security (and $2^{-\kappa}$ simulation error) using $\mathsf{poly}(n, \sigma)$ calls to g.*

Reducing the Number of Calls. Our second primitive improves on efficiency over the first by requiring fewer calls, but requires a preprocessing phase which is implemented using an MPC with identifiability on aborts (in other words, if the protocol fails then all honest players agree on the identity of a corrupted player.) Settling for computational security, such a protocol can be based on the existence of (two-party) oblivious transfer [12].

The protocol for f begins by the having the players run an MPC protocol as above to compute UISS-shares of the output of f. In case the preliminary MPC protocol fails, all players disqualify the player that caused the abort and restart the protocol by using a default value as the input of disqualified players.

We now describe the second primitive which is used to complete the protocol.

- FCR^2 takes as input an n-tuple of UISS shares for a β-bit secret and reconstructs the secret. In case reconstruction succeeds the primitive returns the reconstructed value to all players. If reconstruction fails, the primitive outputs a partition of the players by the lists output by the UISS scheme as in FCR^1.

The protocol for f proceeds by repeatedly interleaving the preliminary (computational) MPC with calls to FCR^2 until an output value is successfully reconstructed by the latter. Each failure results in the honest players disqualifying at least one corrupted player. As before, in each point of the protocol all honest players agree on the identity of disqualified players.

Theorem 7. *Suppose an oblivious transfer protocol exists with computational security parameter λ. Then there is a deterministic, polynomial-time computable n-party functionality g with input and output size $\mathsf{poly}(n, \beta, \kappa)$ such that any polynomial-time computable f with output size β can be realized with full computational security, up to $\mathsf{neg}(\lambda) + 2^{-\kappa}$ simulation error, using at most n calls to g.*

In the full version we give two variants on the above theorem that eliminate the dependence on the output length at the price of increased complexity of the MPC phase, and reduce the number of calls to 1 at the price of increasing the complexity of the primitive exponentially in n.

5.3 Partial Fairness with preprocessing

In this section we briefly sketch how the unanimously identifiable commitments (UIC) primitive can be used with the partially fair MPC protocols of Beimel et al. [2, 1] to eliminate the assumption of cryptographic signatures in the preprocessing model.

Construction with an Off-Line Dealer. The MPC protocols from [2, 1] achieve unconditional security against *fail-stop* adversaries (with a non-negligible error $1/p$) given a trusted preprocessing phase in which a dealer sends some secret information to each player. This information contains the messages each player should send during the protocol, but the choice of which message is sent may depend on the (public) identity of the players that aborted up to this point. To upgrade the security of such a protocol to hold against active adversaries, Beimel et al. rely on digital signatures to ensure that players do not deviate from their designated messages. Our observation is that one could instead rely on the UIC primitive by having the dealer give to the player who should send a message the decommitment information for this message and to all other players the corresponding commitments. Then, if a corrupted player attempts to modify this decommitment information, all honest players will recognize this simultaneously and continue the execution as if this player had aborted.

Note that when considering general MPC in this model (rather than coin-flipping), it may be useful to allow the preprocessing stage to depend on the players' inputs. We refer to such a preprocessing phase as *input dependent preprocessing*. Since we require the outputs of the protocol to be unpredictable in the end of the preprocessing phase,[4] input dependent preprocessing cannot be used to trivially solve the problem by simply delivering the outputs of f to the players.

Theorem 8. *Let $\mathcal{P}$ be an r-round protocol with input dependent preprocessing, which realizes $\mathcal{F}$ with ϵ-security against fail-stop adversaries who can corrupt up to t players. Furthermore, suppose that the online phase of $\mathcal{P}$ has the following structure: in each round, each player sends a subset of the messages it had received in the preprocessing phase, where the identity of this subset can be computed publicly from the pattern of aborts up to this round. Then, there is a protocol $\mathcal{P}'$ with the same features of $\mathcal{P}$ except that it is $(\epsilon + 2^{-\kappa})$-secure against* active *adversaries.*

In the case of randomized functionalities with no inputs, the above theorem does not require the preprocessing to depend on any inputs. In particular, applying the above theorem to the coin-flipping protocol with preprocessing implicit in the construction from [2], we get the following corollary.

Theorem 9. *Assume preprocessing by a trusted off-line dealer. Fix constants n and t such that $t < 2n/3$. Then, for any r, there is an r-round n-party unconditionally secure coin-flipping protocol over point-to-point channels tolerating up to t malicious players with bias $O(1/r)$.*

In the full version we present a variant of our general UIC-based technique which can make the preprocessing phase independent of the inputs. This variant efficiently applies only when the number of players is constant and the input and

[4] More precisely, security in the preprocessing model requires to simulate the adversary's view in the preprocessing phase before invoking the ideal functionality.

output domain of each player is polynomially bounded in the security parameter. Applying this variant to general MPC protocols with $1/p$-security from [1], we obtain the following theorem.

Theorem 10. *Assume preprocessing by a trusted off-line dealer. Let n and t be constants such that $n/2 \leq t < 2n/3$ and $\mathcal{F}$ be a deterministic n-party functionality with input domain bounded by a polynomial $d(\kappa)$ for each player. Then, for any polynomial $p(\kappa)$, there is a polynomial-time r-round $1/p$ secure protocol for $\mathcal{F}$ which tolerates up to t corrupt players with $r = pd^{n \cdot 2^t}$.*

Acknowledgements. We would like to acknowledge the helpful comments and suggestions of the anonymous TCC reviewers, and in particular for pointing out the relevance of the notion of IC Signatures from [25].

References

1. A. Beimel, Y. Lindell, E. Omri, and I. Orlov. 1/p-secure multiparty computation without honest majority and the best of both worlds. In *Crypto '11*, pages 277–296.
2. A. Beimel, E. Omri, and I. Orlov. Protocols for multiparty coin toss with dishonest majority. In *Crypto '10*, pages 538–557.
3. E. F. Brickell and D. R. Stinson. The detection of cheaters in threshold schemes. *SIAM J. Discrete Math.*, 4(4):502–510, 1991.
4. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
5. M. Carpentieri. A perfect threshold secret sharing scheme to identify cheaters. *Designs, Codes and Cryptography*, 5(3):183–187, 1995.
6. B. Chor and E. Kushilevitz. A zero-one law for boolean privacy (extended abstract). In *STOC '89*, pages 62–72.
7. A. Choudhury. Simple and asymptotically optimal t-cheater identifiable secret sharing scheme. *IACR Cryptology ePrint Archive*, 2011:330, 2011.
8. R. Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *STOC '86*, pages 364–369. ACM.
9. R. Cramer, Y. Dodis, S. Fehr, C. Padro, and D. Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In *EUROCRYPT '08*, pages 471–488.
10. S. Fehr and U. M. Maurer. Linear VSS and distributed commitments based on secret sharing and pairwise checks. In *Crypto '02*, pages 565–580, 2002.
11. M. Fitzi, J. A. Garay, U. M. Maurer, and R. Ostrovsky. Minimal complete primitives for secure multi-party computation. *J. Cryptology*, 18(1):37–61, 2005.
12. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC '87*, pages 218–229.
13. S. D. Gordon, Y. Ishai, T. Moran, R. Ostrovsky, and A. Sahai. On complete primitives for fairness. In *TCC '10*.
14. S. D. Gordon and J. Katz. Partial fairness in secure two-party computation. In *EUROCRYPT '10*, pages 157–176.
15. V. Goyal, Y. Ishai, M. Mahmoody, and A. Sahai. Interactive locking, zero-knowledge PCPs, and unconditional cryptography. In *Crypto '10*, pages 173–190.

16. Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In *Crypto '08*, pages 572–591.
17. J. Katz. Universally composable multi-party computation using tamper-proof hardware. In *EUROCRYPT '07*, pages 115–128.
18. J. Kilian. Founding cryptography on oblivious transfer. In *STOC '88*, pages 20–31. ACM.
19. J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in private computations. *SIAM J. Comput. '00*, 29(4):1189–1208.
20. K. Kurosawa, S. Obana, and W. Ogata. t-cheater identifiable (k, n) threshold secret sharing schemes. In *Crypto '95*, pages 410–423, 1995.
21. E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in multi-party private computations. In *FOCS '94*, pages 478–489.
22. R. J. McEliece and D. V. Sarwate. On sharing secrets and Reed-Solomon codes. *Commun. ACM '81*, 24(9):583–584.
23. T. Moran, M. Naor, and G. Segev. An optimally fair coin toss. In *TCC '09*, pages 1–18.
24. S. Obana and T. Araki. Almost optimum secret sharing schemes secure against cheating for arbitrary secret distribution. *ASIACRYPT '06*, pages 364–379.
25. A. Patra, A. Choudhary, and C. Pandu Rangan. Round efficient unconditionally secure multiparty computation protocol. In *INDOCRYPT*, pages 185–199, 2008. Full version in eprint report 2008/399.
26. B. Pfitzmann and M. Waidner. Information-theoretic pseudosignatures and byzantine agreement for t= n/3. *IBM Research Report RZ '96*, 2882.
27. B. Pfitzmann and M. Waidner. Unconditional byzantine agreement for any number of faulty processors. *STACS '92*, pages 337–350.
28. T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *STOC '89*, pages 73–85.
29. P. Rogaway and M. Bellare. Robust computational secret sharing and a unified account of classical secret-sharing goals. In *CCS '07*, pages 172–184.