Secure Multi-Party Computation with Identifiable Abort

Yuval Ishai[1]*, Rafail Ostrovsky[2]**, and Vassilis Zikas[3]***

[1] Computer Science Department, Technion
yuvali@cs.technion.ac.il
[2] Computer Science Department, UCLA
rafail@cs.ucla.edu
[3] Computer Science Department, ETH Zurich
vzikas@inf.ethz.ch

Abstract. Protocols for secure multi-party computation (MPC) that resist a dishonest majority are susceptible to "denial of service" attacks, allowing even a single malicious party to force the protocol to abort. In this work, we initiate a systematic study of the more robust notion of *security with identifiable abort*, which leverages the effect of an abort by forcing, upon abort, at least one malicious party to reveal its identity.

We present the first *information-theoretic* MPC protocol which is secure with identifiable abort (in short ID-MPC) using a correlated randomness setup. This complements a negative result of Ishai et al. (TCC 2012) which rules out information-theoretic ID-MPC in the OT-hybrid model, thereby showing that *pairwise* correlated randomness is insufficient for information-theoretic ID-MPC.

In the standard model (i.e., without a correlated randomness setup), we present the first computationally secure ID-MPC protocol making *black-box* use of a standard cryptographic primitive, namely an (adaptively secure) oblivious transfer (OT) protocol. This provides a more efficient alternative to existing ID-MPC protocols, such as the GMW protocol, that make a non-black-box use of the underlying primitives.

As a theoretically interesting sidenote, our black-box ID-MPC provides an example for a natural cryptographic task that can be realized using a *black-box access* to an OT protocol but cannot be realized unconditionally using an ideal OT oracle.

* Supported by the European Union's Tenth Framework Programme (FP10/2010-2016) under grant agreement no. 259426 ERC-CaC, ISF grant 1361/10, and BSF grant 2012378.

** Work supported in part by NSF grants 09165174, 1065276, 1118126 and 1136174, US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is based upon work supported by 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.

*** Portions of this work were done at UCLA. Work supported in part by the Swiss National Science Foundation (SNF) Ambizione grant PZ00P2_142549.

1 Introduction

Recent advances in secure multiparty computation have led to protocols that compute large circuits in a matter of seconds. Most of these protocols, however, are restricted to provide security against semi-honest adversaries, or alternatively assume an honest majority. A notable exception is the SPDZ line of work [3,16,14,15,34] which tolerates a majority of malicious parties. SPDZ is optimized for the pre-processing model and demonstrates a remarkably fast on-line phase, largely due to the fact that it uses information-theoretic techniques and, thus, avoids costly cryptographic operations. Unfortunately, all these efficient MPC protocols for the case of a dishonest majority are susceptible to the following *denial-of-service (DoS)* attack: even a single malicious party can force an abort without any consequences (i.e., without even being accused of cheating). Although classical impossibility results for MPC prove that abort-free computation is impossible against dishonest majorities, vulnerability to DoS attacks is an issue that should be accounted for in any practical application.

Summary of known results. The seminal works on MPC [47,21,2,9,42] establish tight feasibility bounds on the tolerable number of corruptions for perfect, statistical (aka information-theoretic or unconditional), and computational (aka cryptographic) security. For semi-honest adversaries, unconditionally secure protocols exist if there is an honest majority, or if the parties have access to a complete functionality oracle or other types of setup. An arguably minimal setup is giving the parties (appropriately) correlated random strings before the inputs are known. We refer to this as the *correlated randomness model*.

When there is no honest majority and the adversary is malicious, full security that includes fairness cannot be achieved [12]. Instead, one usually settles for the relaxed notion of *security with abort*: Either the protocol succeeds, in which case every party receives its output, or the protocol aborts, in which case all honest parties learn that the protocol aborted. (Because of the lack of fairness, the adversary can learn its outputs even when the protocol aborts.) The GMW protocol [21,19] realizes this notion of security under standard cryptographic assumptions. Interestingly, this protocol also satisfies the following useful *identifiability* property: upon abort every party learns the identity of some corrupted party. This property is in the focus of our work.

To the best of our knowledge, all protocols achieving this notion of security (e.g., [21,7]) are based on the same paradigm of using public zero-knowledge proofs to detect deviation from the protocol. While elegant and conceptually simple, this approach leads to inefficient protocols that make a non-black-box use of the underlying cryptographic primitives.[4] The situation is even worse in the information-theoretic setting, where an impossibility result from [31] (see

[4] Alternatively, protocols such as the CDN protocol [13] make a use of ad-hoc zero-knowledge proofs based on specific number theoretic intractability assumptions. The disadvantage of these protocols is that they require public-key operations for each gate of the circuit being evaluated, and cannot get around this by using optimization techniques such as efficient OT extension [28].

also [44, Section 3.7]) proves that information-theoretic MPC with identifiable abort is impossible even in the OT-hybrid model, i.e., where parties can make ideal calls to an oblivious transfer (OT) functionality [41].

Our Contributions. We initiate a systematic study of this more robust and desirable notion of *secure MPC with identifiable abort* (ID-MPC). An ID-MPC protocol leverages the effect of an abort by forcing, upon abort, at least one malicious party to reveal its identity. This feature discourages cheaters from aborting, and in many applications allows for full recovery by excluding the identified cheater and restarting the protocol. We provide formal security definitions both in the setting of Universal Composition (UC) [5] and in the stand-alone setting [21,19,4]. Furthermore, we study feasibility and efficiency of ID-MPC in both the information-theoretic and the computational security models.

For the information-theoretic model, we present a general compiler that transforms any MPC protocol which uses correlated randomness to achieve security against semi-honest adversaries into a similar protocol which is secure with identifiable abort against malicious adversaries. As a corollary, we get the first information-theoretic ID-MPC protocol in the correlated randomness model. This protocol complements an impossibility result from [31], which rules out information-theoretic ID-MPC in the OT-hybrid model. Indeed, the insufficiency of OT implies that *pairwise* correlated randomness is not sufficient for information-theoretic ID-MPC, but leaves open the question of whether or not n-wise correlations are, which is answered affirmatively here.

In the computational security model, we present an ID-MPC protocol for realizing sampling functionalities, namely ones that sample and distribute correlated random strings, which only makes a *black-box* use of an (adaptively secure) *OT protocol* and ideal calls to a commitment functionality.[5] Using this protocol for realizing the setup required by the information-theoretic protocol yields the first ID-MPC protocol which makes a black-box use of standard cryptographic primitives. This holds both in the UC framework [5], under standard UC-setups, and in the plain stand-alone model [21,19,4]. Combined with the abovementioned impossibility result from [31], this provides an interesting example for a natural cryptographic task that can be realized using a *black-box* access to an OT protocol but cannot be unconditionally realized using an ideal OT oracle.

Our results demonstrate that ID-MPC is not only the most desirable notion from a practical point of view, but it also has the potential to be efficiently implemented. To this end, one can instantiate our construction with efficient OT protocols from the literature [39,10,36,17].[6] Furthermore, pre-computing the randomness in an off-line phase yields a protocol in the pre-processing model which,

[5] The ideal commitments can be replaced by a black-box use of a commitment protocol, or alternatively realized by making a black-box use of OT [27,38]. The OT protocol can be secure against either semi-honest or malicious adversaries, as these two flavors are equivalent under black-box reductions [24,11].

[6] Our analysis requires the underlying OT to be adaptively secure. Proving the same statement for a static OT protocol is a theoretically interesting open problem. From a practical point of view, however, many instances of adaptively secure OT can

similarly to SPDZ-style protocols, has an information-theoretic online phase. Investigating how our methodology can be fine-tuned towards practice remains an interesting direction for future work. Finally, our protocols can be used to improve the efficiency of a number of protocols in the fairness-related literature, e.g., [29,18,26,37,48,22,1], as these works implicitly use ID-MPC (typically instantiated by GMW) to realize a sampling functionality.

Comparison to Existing Work. Our information-theoretic protocol can be seen as a new *feasibility* result, since the current literature contains no (efficient or inefficient) information-theoretic ID-MPC protocol from correlated randomness. Similarly, our computational protocol can also be seen as a "second-order" feasibility result, since this is the first ID-MPC protocol making *black-box* use of a standard cryptographic primitive. Notwithstanding, much of the motivation for considering black-box constructions in cryptography is derived from the goal of practical efficiency, and indeed the most practical protocols today (whether Yao-based or GMW-based) are black-box protocols that do not need to know the "code" of the underlying cryptographic primitives.

2 The Model

We prove our security statements in the universal composition (UC) framework of Canetti [5]: in a nutshell, a protocol π (securely) UC realizes a functionality $\mathcal{F}$ if for any adversary $\mathcal{A}$ attacking π there exists an ideal adversary, the simulator $\mathcal{S}$, that makes an ideal evaluation of $\mathcal{F}$ indistinguishable from a protocol execution with $\mathcal{A}$ in the eyes any environment $\mathcal{Z}$. When $\mathcal{Z}$, $\mathcal{A}$, and $\mathcal{S}$ are polynomially bounded we say that the protocol realizes $\mathcal{F}$ (with computational security); otherwise, when $\mathcal{Z}$, $\mathcal{A}$, and $\mathcal{S}$ are unbounded, we say that the protocol *unconditionally* realizes $\mathcal{F}$ (with information-theoretic security).

For simplicity we restrict our description to computation of non-reactive functionalities, also known as *secure function evaluation (SFE)*. (The general case can be reduced to this case by using a suitable form of secret sharing [31] for maintaining the secret state of the reactive functionality.) Moreover, we describe our protocols as *synchronous protocols*, i.e., round-based protocols where messages sent in some round are delivered by the beginning of the next round; such protocols can be executed in UC as demonstrated in [33,35]. The advantage of such a "synchronous" description is dual: first, it yields simpler descriptions of functionalities and protocols; indeed, because the parties are aware of the round in which each message should be sent/received, we can avoid always explicitly writing all the message/protocol IDs in the descriptions. Second, it is compatible with the protocol description in the stand-alone model of computation [20,4], which allows us to directly translate our results into that model.

Our protocols assume n parties from the set $\mathcal{P} = \{p_1, \ldots, p_n\}$. We prove our results for a non-adaptive adversary who actively corrupts parties *at the beginning* of the protocol execution, but our results can be extended to the adaptive

be efficiently implemented from few such instances in the (programmable) random oracle model [28,36].

case.[7] Our results are with respect to an (often implicit) security parameter k, where we use the standard definition of negligible and overwhelming from [19].

Correlated Randomness as a Sampling Functionality. Our protocols are in the *correlated randomness* model, i.e., they assume that the parties initially, before receiving their inputs, receive appropriately correlated random strings. In particular, the parties jointly hold a vector $\boldsymbol{R} = (R_1, \ldots, R_n) \in (\{0,1\}^*)^n$, where p_i holds R_i, drawn from a given efficiently samplable distribution $\mathcal{D}$. This is, as usual, captured by giving the parties initial access to an ideal functionality $\mathcal{F}_{\mathsf{Corr}}^{\mathcal{D}}$, known as a *sampling functionality*, which, upon receiving a default input from any party, samples $\boldsymbol{R}$ from $\mathcal{D}$ and distributes it to the parties. Hence, a protocol in the correlated randomness model is formally an $\mathcal{F}_{\mathsf{Corr}}^{\mathcal{D}}$-hybrid protocol.

Information-Theoretic Signatures. Our protocols use information-theoretic (i.t.) signatures [45,43,46] to commit a party to messages it sends. Roughly speaking, these are information-theoretic analogues to standard digital signatures, i.e., they allow some party p_i, the *signer*, to send a message m to a party p_j, the *receiver*, along with a string σ that we refer to as the *signature*, such that the receiver can at a later point publicly open σ and prove to every party that the message m was indeed sent from p_i. Note that in order to achieve i.t. security the verification key cannot be publicly known. Rather, in i.t. signatures, the signer has a signing key $\mathtt{sk}$ and every party $p_i \in \mathcal{P}$ holds a different private verification key $\mathtt{vk}_i$ corresponding to $\mathtt{sk}$.

In our protocols different (independent) signing keys are used for each signature. In this case, i.t. signatures provide the following guarantees with overwhelming probability (against an unbounded adversary): *(completeness)* A signature with the correct singing key will be accepted by any honest verifier in $\mathcal{P}$; *(unforgeability)* the adversary cannot come up with a signature that will be accepted by some (honest) verifier without knowing the signing key; *(consistency)* an adversarial signer cannot come up with a signature that will be accepted by some honest verifier and rejected by another.

3 Security with Identifiable Abort

We put forward the notion of *secure multi-party computation with identifiable abort*, also referred to as *Identifiable MPC (ID-MPC)* which allows the computation to fail (abort), but ensures that when this happens every party is informed about it, and they also agree on the index i of some corrupted party $p_i \in \mathcal{P}$ (we say then that *the parties abort with* p_i). More concretely, for an arbitrary functionality $\mathcal{F}$, we define $[\mathcal{F}]_{\perp}^{\mathtt{ID}}$ to be the corresponding functionality with identifiable abort, which behaves as $\mathcal{F}$ with the following modification: upon receiving from the simulator a special command (abort, p_i), where $p_i \in \mathcal{P}$ is a corrupted party (if p_i is not corrupted then $[\mathcal{F}]_{\perp}^{\mathtt{ID}}$ ignores the message), $[\mathcal{F}]_{\perp}^{\mathtt{ID}}$ sets the output of all (honest) parties to (abort, p_i).

[7] In fact, some of our protocols use optimizations tailored to proving adaptive security.

Definition 1. *Let $\mathcal{F}$ be a functionality and $[\mathcal{F}]^{\mathtt{ID}}_{\perp}$ be the corresponding functionality with identifiable abort. We say that* a protocol π securely realizes $\mathcal{F}$ with identifiable abort *if π securely realizes the functionality $[\mathcal{F}]^{\mathtt{ID}}_{\perp}$.*

The UC composition theorem extends in a straightforward manner to security with identifiable abort. The concrete composition statement can be found in the full version.

4 Unconditional ID-MPC from Correlated Randomness

In this section we describe our unconditionally secure identifiable MPC protocol in the correlated randomness model. In fact, our result is more general, as we provide a compiler that transforms any given unconditionally secure protocol in the semi-honest correlated randomness model into an unconditionally secure ID-MPC protocol in the (malicious) correlated randomness model. Although the correlated randomness provided by the setup in the malicious protocol is different than the semi-honest, the latter can be obtained from the former by an efficient transformation. Informally, our statement can be phrased as follows:

> Let π_{sh} be an $\mathcal{F}_{\mathsf{Corr}}^{\mathcal{D}}$-hybrid protocol (for an efficiently computable distribution $\mathcal{D}$), which *unconditionally* UC realizes a functionality $\mathcal{F}$ in the presence of a *semi-honest* adversary. Then there exists a compiler turning π_{sh} into an $\mathcal{F}_{\mathsf{Corr}}^{\mathcal{D}'}$-hybrid protocol (for an appropriate efficiently computable distribution $\mathcal{D}'$), which *unconditionally* UC realizes $\mathcal{F}$ *with identifiable abort* (in the malicious model).

Overview of the Compiler. We start by providing a high-level overview of our compiler. As is typical, the semi-honest protocol π_{sh} which we compile works over standard point-to-point (insecure) channels. Furthermore, without loss of generality (see Section 4.3) we assume that π_{sh} is *deterministic*.

Let $\boldsymbol{R}^{\mathtt{sh}} = (R_1^{\mathtt{sh}}, \ldots, R_n^{\mathtt{sh}})$ denote the setup used by the semi-honest protocol π_{sh} (i.e., each p_i holds string $R_i^{\mathtt{sh}}$). The setup for the compiled protocol distributes $\boldsymbol{R}^{\mathtt{sh}}$ to the parties, and commits every party to its received string. Subsequently, the parties proceed by, first, committing to their inputs and, then, executing their π_{sh}-instructions in a publicly verifiable manner: whenever, p_i would send a message m in π_{sh}, in the compiled protocol p_i broadcasts m and publicly proves, in zero-knowledge, that the broadcasted message is consistent with his committed input and setup string $R_i^{\mathtt{sh}}$. For the above approach to work for unbounded adversaries and allow for identifiability, we need the commitment scheme and the associated zero-knowledge proofs to be unconditionally secure and failures to be publicly detectable. We construct such primitives relying on appropriately correlated randomness in Sections 4.1 and 4.2, respectively.

4.1 Commitments with Identifiable Abort

In this section we provide a protocol which unconditionally UC realizes the standard (one-to-many) multi-party commitment functionality $\mathcal{F}_{\mathrm{COM}}$ with identifiable

abort. $\mathcal{F}_{\text{COM}}$ allows party $p_i \in \mathcal{P}$, the *committer*, to commit to a message m and later on publicly open m while guaranteeing the following properties: (hiding) no party in $\mathcal{P} \setminus \{p_i\}$ receives any information on m during the commit phase; (binding) at the end of the commit phase a message m' is fixed (where $m' = m$ if the committer is honest), such that only m' might be accepted in the reveal phase (and m' is always accepted when the committer is honest).

Our protocol Π_{COM} which i.t. securely realizes $\mathcal{F}_{\text{COM}}$ with identifiable abort assumes the following correlated-randomness setup: for p_i to commit to a value $m \in \{0,1\}^*$, p_i needs to hold a uniformly random string $r \in \{0,1\}^{|m|}$ along with an information-theoretic signature σ on r, where every party in $\mathcal{P}$ holds his corresponding verification key (but no party, not even p_i, gets to learn the signing key). Given the above setup, p_i can commit to m by broadcasting $y = m \oplus r$. To, later on, open the commitment y, p_i broadcasts r along with the signature σ, where every party verifies the signature and outputs $m = y \oplus r$ if it is valid, otherwise aborts with p_i (i.e., outputs (abort, p_i)).

The hiding property of Π_{COM} follows from the fact that r is uniformly random. Moreover, the unforgeability of the signature scheme ensures that the commitment is binding and publicly verifiable. Finally, the completeness of the scheme ensures that the protocol aborts only when the committer p_i is corrupted. Additionally, same as all UC commitments, the above scheme is *extractable*, i.e., the simulator of a corrupted committer can learn, already in the commit phase, which message will be opened so that he can input it to the functionality, and *equivocal*, i.e., the simulator of a corrupted receiver can open a commitment to any message of his choice.[8] Taking a glimpse at the proof both properties follow from the fact that the simulator controls the setup: knowing r allows the simulator to extract m from the broadcasted message, whereas knowing the signing key $\mathtt{sk}$ allows him to generate a valid signature/opening to any message.

Theorem 1. *The protocol Π_{COM} unconditionally UC realizes the functionality $\mathcal{F}_{\text{COM}}$ with identifiable abort.*

4.2 Setup-Commit-Then-Proof

Next we present a protocol which allows the parties receiving random strings (drawn from some joint distribution $\mathcal{D}$) to publicly prove, in zero-knowledge, that they use these strings in a protocol. Our protocol implements the *Setup-Commit-then-Prove* functionality $\mathcal{F}_{\text{SCP}}$ which can be viewed as a modification of the Commit-then-Prove functionality from [7] restricting the committed witnesses to be distributed by the setup instead of being chosen by the provers. More concretely $\mathcal{F}_{\text{SCP}}$ works in two phases: in a first phase, it provides a string/witness R_i to each $p_i \in \mathcal{P}$, where $\boldsymbol{R} = (R_1, \ldots, R_n)$ is drawn from $\mathcal{D}$; in a second phase, $\mathcal{F}_{\text{SCP}}$ allows every party p_i to prove q-many NP statements of the type

[8] In [31] a primitive called *unanimously identifiable commitments* (UIC) was introduced for this purpose, but the definition of UIC does not guarantee all the properties we need for UC secure commitments.

$\mathcal{R}(x, R_i) = 1$ for the same publicly known NP relation $\mathcal{R}_i$ and the witness R_i received from the setup, but for potentially different (public) strings x.

In the remainder of this section we describe a protocol which unconditionally securely realizes the setup-commit-then-proof functionality $\mathcal{F}_{\text{SCP}}$ in the correlated randomness model. To this direction, we first show how to realize the sigle-use version of $\mathcal{F}_{\text{SCP}}$, denoted as $\mathcal{F}_{1\text{SCP}}$, and then use the UC composition with joint state theorem (JUC) [8] to derive a protocol for $\mathcal{F}_{\text{SCP}}$. The functionality $\mathcal{F}_{1\text{SCP}}$ works exactly as $\mathcal{F}_{\text{SCP}}$ with the restriction that it allows a prover $p \in \mathcal{P}$ to do *a single* (instead of q-many) proofs for a witness w of a given NP relation $\mathcal{R}$.

Our protocol for realizing the functionality $\mathcal{F}_{1\text{SCP}}$ with identifiable abort uses the idea of "MPC in the head" [25,30,32]. In particular, let $\mathcal{F}_{\mathsf{D}}$ denote the $(n+1)$-party (reactive) functionality among the players in $\mathcal{P}$ and a special player p_D, the *dealer*, which works as follows: In a first phase, $\mathcal{F}_{\mathsf{D}}$ receives a message $w \in \{0,1\}^{\text{poly}(k)}$ from p_D and forwards w to $p \in \mathcal{P}$. In a second phase, p sends x to $\mathcal{F}_{\mathsf{D}}$, which computes $b := \mathcal{R}(x, w)$ and outputs (b, x) to every $p_j \in \mathcal{P} \setminus \{p_D\}$. Clearly, any protocol in the plain model which unconditionally realizes $\mathcal{F}_{\mathsf{D}}$ with an honest dealer p_D, where p_D does not participate in the second phase, can be turned into a protocol which securely realizes $\mathcal{F}_{1\text{SCP}}(\mathcal{P}, \mathcal{D}, \mathcal{R}, p)$ in the correlated randomness model. Indeed, one needs to simply have the corresponding sampling functionality play the role of p_D (where w is drawn from $\mathcal{D}$). In the following we show how to design such a protocol using the idea of player-simulation [25].

Let $\Pi_{(n+1,m),t}$ be a protocol which perfectly securely (and robustly) realizes $\mathcal{F}_{\mathsf{D}}$ in the client-server model [25,30,32], among the clients $\mathcal{P} \cup \{p_D\}$ and an additional m servers. Such a protocol exists assuming $t < m/3$ servers are corrupted [2]. For simplicity, assume that $\Pi_{(n+1,m),t}$ has the following properties, which are consistent to how protocols from the literature, e.g., [2], would realize functionality $\mathcal{F}_{\mathsf{D}}$ in the client-server setting: (i) for computing the first phase of $\mathcal{F}_{\mathsf{D}}$, $\Pi_{(n+1,m),t}$ has p_D share his input w among the m servers with a secret sharing scheme that is perfectly *t-private* (the shares of any t servers leak no information on w) and perfectly *t-robust* (the sharing can be reconstructed even when up to t cheaters modify their shares), and, also p_D hands *all* the shares to p (ii) p_D does not participate in the second phase of $\Pi_{(n+1,m),t}$ (this is wlog as p_D is a client with no input or output in this second phase), and (iii) the output $(\mathcal{R}(x, w), x)$ is publicly announced (i.e., is in the view of every server at the end of the protocol).

Assuming p_D is honest, a protocol Π_{n+1} for unconditionally realizing $\mathcal{F}_{\mathsf{D}}$ with identifiable abort (among only the players in $\mathcal{P} \cup \{p_D\}$) can be built based on the above protocol $\Pi_{(n+1,m),t}$ as follows: for the first phase, p_D generates shares of a t-robust and t-private sharing of w as he would do in $\Pi_{(n+1,m),t}$ and sends them to p. In addition to sending the shares, p_D commits p to each share by sending him an i.t. signature on it and distributing the corresponding verification keys to the players in $\mathcal{P}$. For the second phase, p emulates in his head the second phase of the execution of $\Pi_{(n+1,m),t}$ among m virtual servers $\hat{p}_1, \ldots, \hat{p}_m$ where each server has private input his share, as received from p_D in the first phase, and a public input x (the same for all clients); p publicly commits to the view of

each server. Finally, the parties in $\mathcal{P} \setminus \{p\}$ challenge p to open a random subset $\mathcal{J} \subseteq [m]$ of size t of the committed views and announce the corresponding input-signatures which p received from p_D. If the opened views are inconsistent with an accepting execution of $\Pi_{(n+1,m),t}$ on input x and the committed shares—i.e., some output is 0, or some opening fails, or some signature does not verify for the corresponding (opened) private input, or for some pair of views the incoming messages do not match the outgoing messages—then the parties abort with p.

The security of the protocol Π_{n+1} is argued similarly to [30, Theorem 4.1]: on the one hand, when p is honest then we can use the simulator for $\Pi_{(n+1,m),t}$ to simulate that views of the parties in $\mathcal{J}$. The perfect t-security of $\Pi_{(n+1,m),t}$ and the t-privacy of the sharing ensures that this simulation is indistinguishable from the real execution. On the other hand, when p is corrupted, then we only need to worry about correctness. Roughly, correctness is argued as follows: if there are at most $t < m/3$ incorrect views, then the t-robustness of $\Pi_{(n+1,m),t}$ and of the sharing ensures that the output in any of the other views will be correct; by a standard counting argument we can show that the probability that some of these views is opened is overwhelming when $m = O(k)$. Otherwise, (i.e., if there are more than t-incorrect views) then with high probability a pair of such views will be opened and the inconsistency will be exposed.

To derive, from Π_{n+1}, a protocol for $\mathcal{F}_{\text{1SCP}}(\mathcal{P}, \mathcal{D}, \mathcal{R}, p)$ in the correlated randomness model, we have the sampling functionality, $\mathcal{F}_{\mathsf{Corr}}^{\text{1SCP}}$ play the role of the dealer p_D. In addition to the committed shares, $\mathcal{F}_{\mathsf{Corr}}^{\text{1SCP}}$ generates the necessary setup enabling any prover $p \in \mathcal{P}$ to commit to the m (virtual) servers' views in the second phase of the protocol Π_{n+1}. Furthermore, to simplify the description, we also have $\mathcal{F}_{\mathsf{Corr}}^{\text{1SCP}}$ create a "coin-tossing setup" which players in $\mathcal{P}$ can use to sample the random subset $\mathcal{J} \in [m]$ of views to be opened: $\mathcal{F}_{\mathsf{Corr}}^{\text{1SCP}}$ hands to each $p_j \in \mathcal{P}$ a random string c_j and commits p_j to it; the coin sequence c for choosing $\mathcal{J}$ is then computed by every p_j opening c_j and taking $c = \oplus_{j=1}^{n} c_j$. In the following we give a detailed description of the protocol Π_{1SCP} for implementing $\mathcal{F}_{\text{1SCP}}$, where we denote by $\langle w \rangle = (\langle w \rangle^1, \ldots, \langle w \rangle^m)$ a perfectly t-private and t-robust secret sharing of a given value w among players in some $\hat{\mathcal{P}} = (\hat{p}_1, \ldots, \hat{p}_m)$ (e.g., the sharing from [2] which is based on bivariate polynomials), where $\langle w \rangle^i$ denotes the ith share of $\langle w \rangle$, i.e., the state of the (virtual) server $\hat{p}_i$ after the sharing is done.

Theorem 2. *Let $\Pi_{(n+1,m),t}$ be a protocol as described above among $n+1$ clients and $m = O(k)$ servers which perfectly securely (and robustly) realizes the functionality $\mathcal{F}_{\mathsf{D}}$ in the presence of $t < m/3$ corrupted servers. The $(\mathcal{F}_{\mathsf{Corr}}^{\text{1SCP}}(\mathcal{P}, \mathcal{D}, m, t, \mathcal{R})$-hybrid) protocol $\Pi_{\text{1SCP}}(\mathcal{P}, \mathcal{D}, \mathcal{R}, m, t, p)$ unconditionally securely realizes the functionality $\mathcal{F}_{\text{1SCP}}(\mathcal{P}, \mathcal{D}, \mathcal{R}, p)$ with identifiable abort.*

The multiple-proof extension of $\mathcal{F}_{\text{1scp}}$. In order to realize functionality $\mathcal{F}_{\text{SCP}}$ we need to extend $\mathcal{F}_{\text{1SCP}}$ to distribute a vector $\boldsymbol{R} = (R_1, \ldots, R_n)$ of witnesses, one for each party, (instead of only one witness) sampled from some efficient distribution $\mathcal{D}$, and allow every $p_i \in \mathcal{P}$ to prove up to q statements of the type

Protocol $\Pi_{\text{1SCP}}(\mathcal{P}, \mathcal{D}, m, t, \mathcal{R}, p)$

Setup-Commit Phase: To obtain the appropriate setup, prover p sends $(\mathsf{CorrRand}, p)$ to the sampling functionality $\mathcal{F}_{\mathsf{Corr}}^{\text{1SCP}}(\mathcal{P}, \mathcal{D}, m, t, \mathcal{R})$, which distributes the following random strings and signatures (where every $p_j \in \mathcal{P}$ receives the corresponding verification keys):

- The prover p receives a sharing $\langle w \rangle = (\langle w \rangle^1, \ldots, \langle w \rangle^m)$ of w along with corresponding signatures $\sigma(\langle w \rangle^1), \ldots, \sigma(\langle w \rangle^m)$ and (privately) outputs $(\mathsf{witness}, w)$.
- Every $p_i \in \mathcal{P}$ receives the challenge-string c_i along with a corresponding signature $\sigma(c_i)$.
- The prover also receives random strings $v_1, \ldots, v_m$ along with corresponding signatures $\sigma(v_1), \ldots, \sigma(v_m)$ to use for committing to the server's views in $\Pi_{(n+1,m),t}$.

Prove Phase: Upon p receiving his input $(\mathsf{ZK\text{-}prover}, x)$ the following steps are executed:

1. If $\mathcal{R}(x, w) = 0$ then p broadcasts $(\mathsf{not\text{-}verified}, p)$ and every party halts with output $(\mathsf{not\text{-}verified}, p)$. Otherwise, p broadcasts $(\mathcal{R}, x)$.
2. p emulates in his head the second phase of protocol $\Pi_{(n+1,m),t}$ where each server $\hat{p}_j \in \hat{\mathcal{P}} = \{\hat{p}_1, \ldots, \hat{p}_m\}$ has private input $\langle w \rangle^j$ and public input x.
3. For each $\hat{p}_j \in \hat{\mathcal{P}}$, p commits, by invocation of protocol $\Pi_{\text{COM}}(\mathcal{P})$, to the view $\text{VIEW}_j \in \{0,1\}^{\mathbb{V}_j}$ of $\hat{p}_j$ in the above emulated execution using v_j from his setup.
4. For each $p_i \in \mathcal{P}$: p_i announces the random string c_i and the corresponding signature $\sigma(c_i)$ and every $p_j \in \mathcal{P}$ verifies, using his corresponding verification keys, validity of the signatures and aborts with p_i in case the check fails.
5. The parties compute $c = \sum_{i=1}^{n} c_i$ and use it as random coins to sample a random t-size set $\mathcal{J} \subseteq [m]$.
6. For each $j \in \mathcal{J}$: p opens the commitment to VIEW_j and announces the signature $\sigma(\langle w \rangle^j)$. If any of the openings fails or any of the announced signatures is not valid for the input-share appearing in the corresponding view, then the protocol aborts with p_i.
7. Otherwise, the parties check that the announced views are consistent with an execution of protocol $\Pi_{(n+1,m),t}$ with the announced inputs in which the (global) output is 1, i.e., they check that in all the announced views the output equals 1 and all signatures are valid, and that for all pairs $(j,k) \in \mathcal{J}^2$: the incoming messages in $\hat{p}_j$'s view match the outgoing messages in $\hat{p}_k$'s view. If any of these checks fails then the protocol aborts with p_i, otherwise, every party outputs $(\mathsf{verified}, x, p)$.

$\mathcal{R}(R_i, x)$ for potentially different public inputs x. The corresponding sampling functionality is derived as follows: it first samples $\boldsymbol{R}$ and subsequently it emulates, for each $p_i \in \mathcal{P}$, q independent setups for Π_{1SCP} (for the same random value R_i and relation $\mathcal{R}_i$). Given such a sampling functionality the protocol Π_{SCP} for unconditionally securely realizing $\mathcal{F}_{\text{SCP}}$ with identifiable abort is straight-forward: The parties receive the random strings $R_1, \ldots, R_n$ along with q proof setups for

each party. Then, for each invocation of the prove phase, party p_i executes the prove phase of protocol Π_{1SCP} using the corresponding proof setup.[9]

Theorem 3. *Protocol $\Pi_{\text{SCP}}(\mathcal{P}, \mathcal{D}, \mathcal{R}, q)$ unconditionally securely realizes the functionality $\mathcal{F}_{\text{SCP}}(\mathcal{P}, \mathcal{D}, \mathcal{R}, q)$ with identifiable abort.*

The proof follows from the security of Π_{1SCP} by a direct application of the universal composition with joint state (JUC) theorem [8].

4.3 The "Semi-honest to Malicious with Abort" Compiler

We are now ready to describe our main compiler, denoted as $\text{C}(\cdot)$ which compiles any given protocol π_{sh} secure in the semi-honest model using (only) correlated randomness into a protocol $\text{C}(\pi_{sh})$ which is secure with abort in the (malicious) correlated randomness model.[10]

We make the following simplifying assumptions on the semi-honest protocol π_{sh} which are without loss of generality, since all existing semi-honest protocols in the correlated randomness model can be trivially turned to satisfy them:

- We assume that π_{sh} has a known (polynomial) upper bound $\texttt{Rnd}_{\pi_{sh}}$ on the number of rounds, where in each round every party sends a single message.
- We assume that π_{sh} is *deterministic*. Any π_{sh} can be turned into such by having the setup include for each $p_i \in \mathcal{P}$ a uniformly random and independent string r_i that p_i uses as his coins.
- Finally, we assume that π_{sh} starts off by having every party send to all parties a one-time pad encryption of his input x_i using as key the first $|x_i|$ bits from r_i (those bits are not reused). Clearly, this modification does not affect the security of π_{sh} as the simulator can easily simulate this step by broadcasting a random string. Looking ahead in the proof, this will allow the simulator to extract the corrupted parties' inputs.

The compiler $\text{C}(\pi_{sh})$ uses the protocol Π_{SCP} as follows: Denote by $R^{\texttt{sh}} = (R_1^{\texttt{sh}}, \ldots, R_n^{\texttt{sh}})$ the setup used by π_{sh} and by $\mathcal{D}^{\texttt{sh}}$ the corresponding distribution. Let also $\mathcal{R}_{\pi_{sh},i}$ denote the relation corresponding to p_i's next message function. More concretely, if $h_{\pi_{sh},i} \in \{0,1\}^*$ denotes the history of messages seen by p_i and m is a message, then $\mathcal{R}_{\pi_{sh},i}((h_{\pi_{sh},i}, m), R_i) = 1$ if m is the next message of p_i in an execution with history $h_{\pi_{sh},}$ and setup R_i, otherwise $\mathcal{R}_{\pi_{sh},i}((h_{\pi_{sh},i}, m), R_i) = 0$. The compiled protocol $\text{C}(\pi_{sh})$ starts by executing the setup-commit phase of protocol $\Pi_{\text{SCP}}(\mathcal{P}, \mathcal{D}^{\texttt{sh}}, \boldsymbol{\mathcal{R}} = (\mathcal{R}_{\pi_{sh},1}, \ldots, \mathcal{R}_{\pi_{sh},n}), \texttt{Rnd}_{\pi_{sh}})$. Subsequently, every $p_i \in \mathcal{P}$ executes his π_{sh} instructions, where in each round instead of sending its message m over the point-to-point channel, p_i broadcasts m and proves, using the proof phase of protocol Π_{SCP}, that $\mathcal{R}_{\pi_{sh},n}((h_{\pi_{sh},}, m), R_i) = 1$. If Π_{SCP} aborts with some p_i then our compiler also aborts with p_i. Otherwise, the security of Π_{SCP}

[9] Recall that we implicitly assume that all messages generated from the setup have unique identifiers so that the parties know which ones to use for which proof.

[10] Note that $\text{C}(\pi_{sh})$ uses broadcast which can be trivially realized by a protocol assuming appropriate correlated randomness, e.g., [40].

ensures that every p_i followed π_{sh} for the given setup; therefore, security of our compiler follows from the security of π_{sh}. Note that the corresponding sampling functionality for $\mathrm{C}(\pi_{sh})$ is computable in time polynomial in the running time of the sampling functionality $\mathcal{F}_{\mathsf{Corr}}^{\mathcal{D}^{\mathrm{sh}}}$ for protocol π_{sh}.

Theorem 4. *Let π_{sh} be a protocol as above which unconditionally UC realizes a functionality $\mathcal{F}$ in the presence of a semi-honest adversary in the $\mathcal{F}_{\mathsf{Corr}}^{\mathcal{D}^{\mathrm{sh}}}$-hybrid (correlated randomness) model. Then the compiled protocol $\mathrm{C}(\pi_{sh})$ unconditionally UC realizes the functionality $\mathcal{F}$ with identifiable abort in the presence of a malicious adversary in the $\mathcal{F}_{\mathsf{Corr}}^{\mathrm{SCP}}$-hybrid (correlated randomness) model.*

Note that any (semi-honest) OT-hybrid protocol can be cast as a protocol in the correlated randomness model by precomputing the OT. Hence, by instantiating π_{sh} with any semi-honest OT hybrid protocol. e.g., [20], we obtain the following corollary.

Corollary 1. *There exists a protocol which unconditionally UC realizes any well-formed [7] multi-party functionality with identifiable abort.*

The question of feasibility of unconditional security with identifiable abort from correlated randomness has been open even in the simpler *standalone* model [21,19,4]. As a corollary of Theorem 4 one can derive a positive statement also for that model.

Corollary 2 (Stand-alone security with identifiable abort). *There exists a protocol which unconditionally securely evaluates any given function f with identifiable abort in the stand alone correlated randomness model.*

5 SFE Using Black-box OT

In this section, we provide a generic MPC protocol which is (computationally) secure with identifiable abort making black-box use of an (adaptively) secure UC protocol for one-out-of two oblivious transfer $\mathcal{F}_{\mathsf{OT}}$ (see [39] for a formal description) in the Common Reference String (CRS) model.

The high-level idea of our construction is the following: as we have already provided an unconditional implementation of ID-MPC based (only) on correlated randomness, it suffices to provide a protocol $\Pi_{\mathtt{Corr}}^{\mathrm{CSP}}$ with the above properties for implementing the corresponding sampling functionality $\mathcal{F}_{\mathsf{Corr}}^{\mathrm{SCP}}$. Indeed, given such a protocol $\Pi_{\mathtt{Corr}}^{\mathrm{CSP}}$, we can first use it to compute the setup needed for $\mathrm{C}(\pi_{sh})$ (for any appropriate semi-honest protocol π_{sh}, e.g., the one from [21]) and then use π_{sh} to evaluate any given functionality; if either the setup generation or π_{sh} aborts with some p_i then the construction also aborts with p_i.

In the remainder of this section we describe $\Pi_{\mathtt{Corr}}^{\mathcal{F}_{\mathsf{Corr}}^{\mathrm{SCP}}}$. In fact, we provide a protocol $\Pi_{\mathtt{Corr}}^{\mathcal{D}}$ which allows to implement any sampling functionality $\mathcal{F}_{\mathsf{Corr}}^{\mathcal{D}}$ for a given efficiently computable distribution $\mathcal{D}$. The key idea behind our construction in the following: as the functionality $\mathcal{F}_{\mathsf{Corr}}^{\mathcal{D}}$ receives no (private) inputs from

the parties, we can have every party commit to its random tape, and then attempt to realize $\mathcal{F}_{\mathsf{Corr}}^{\mathcal{D}}$ by a protocol which is secure with (non-identifiable) abort; if the evaluation aborts then the parties open the commitments to their random tapes and use these tapes to detect which party cheated. Note that, as the parties have no private inputs, announcing their views does not violate privacy of the computation.

For the above idea to work we need to ensure that deviation from the honest protocol can be consistently detected by every party (upon opening the committed random coins). Therefore, we define the following $\mathcal{P}$-*verifiability* property. For any given execution of a protocol Π, we say that a party p_i *correctly executed* Π *with respect to* (x_i, r_i) *(up to round* ρ*) in the CRS model* if p_i sent all his messages as instructed by Π on this input x_i, random coins r_i and the common reference string C. Let Π be a protocol in the CRS model which starts by having every party commit to its random tape. Π is $\mathcal{P}$-*verifiable* if there exists a deterministic polynomial algorithm $\mathcal{D}$, called *the detector*, with the following property: given the CRS, the inputs of the parties, their committed randomness, and the view of any honest p_j, $\mathcal{D}$ outputs the identity of a party $p_i \in \mathcal{P}$ who did not correctly execute Π (if such a party exists).

In the remainder of this section we provide the details of our protocol. As our protocols makes black-box use of a UC secure 12OT protocol in the CRS model, for it to be $\mathcal{P}$-verifiable the underlying 12OT protocol needs to also be $\mathcal{P}$-verifiable. Therefore, in the following, first, we show how to obtain from any given OT protocol $\Pi_{\mathtt{OT}}$ a $\mathcal{P}$-verifiable OT protocol $\Pi_{\mathtt{VOT}}$ (making black-box use of $\Pi_{\mathtt{OT}}$), and, subsequently, we show how to use $\Pi_{\mathtt{VOT}}$ to transform an OT-hybrid SFE protocol into a $\mathcal{P}$-verifiable SFE protocol in the CRS model. Finally, at the end of the current section, we show how to use our $\mathcal{P}$-verifiable SFE protocol to implement any sampling functionality $\mathcal{F}_{\mathsf{Corr}}^{\mathcal{D}}$ with identifiable abort making black-box use of $\Pi_{\mathtt{OT}}$. **$\mathcal{P}$-Verifiable OT.** Let $\Pi_{\mathtt{OT}}$ be a (two-party) protocol which adaptively UC securely realizes $\mathcal{F}_{\mathsf{OT}}$, among parties p_1 and p_2 in the CRS model (e.g., [10,39]). For $i \in \{1, 2\}$ denote by $f^i_{\Pi_{\mathtt{OT}}}$ the next message function of p_i defined as follows: let VIEW_i be the view of party p_i at the beginning of round ρ in an execution of $\Pi_{\mathtt{OT}}$;[11] then $f^i_{\Pi_{\mathtt{OT}}}(\text{VIEW}_i) = m$ is the message which p_i sends in round ρ of protocol $\Pi_{\mathtt{OT}}$, given that his current view is VIEW_i (if ρ is the last round, then, by default, $m = (\mathsf{out}, y)$, where y is p_i's output). Observe that $f_{\Pi_{\mathtt{OT}}}$ is a deterministic function. Without loss of generality, assume that protocol $\Pi_{\mathtt{OT}}$ has a known number of rounds $\mathtt{Rnd}_{\Pi_{\mathtt{OT}}}$, where in each round only one of the parties p_1 and p_2 sends a message (from $\{0,1\}^k$). Let, also, $\mathcal{F}_{\mathsf{OT}}^{\mathcal{P}}$ denote the multi-party extension of $\mathcal{F}_{\mathsf{OT}}$, in which parties other than p_1 and p_2 provide a default input and receive a default output, i.e., $\mathcal{F}_{\mathsf{OT}}^{\mathcal{P}}$ corresponds to the function $f_{\text{OT}}^{\mathcal{P}}((x_0, x_1), b, \lambda, \ldots, \lambda) = (\perp, x_b, \perp, \perp)$. We describe a multi-party $\mathcal{P}$-verifiable protocol $\Pi_{\mathtt{VOT}}$ which securely realizes the functionality$\mathcal{F}_{\mathsf{OT}}^{\mathcal{P}}$.

The protocol $\Pi_{\mathtt{VOT}}$ works as follows: initially, every party commits to its random tape. Subsequently, the parties execute their $\Pi_{\mathtt{OT}}$ instructions with the

[11] Recall that VIEW_i consists of the inputs and randomness of p_i along with all messages received up to round r.

following modification: whenever, for $i, j \in \{1, 2\}$, p_i is to send a message $m \in \{0, 1\}^k$ to p_j, he chooses the first k unused bits from his random tape (denote by K the string resulting by concatenating these bits), broadcast a one-time pad encryption $c = m \oplus K$ of m with key K, and *privately* opens the corresponding commitments towards p_j. If the opening fails then p_j publicly complains and p_i replies by broadcasting K; p_j recovers m by decrypting c. Clearly, the above modification does not affect the security of $\Pi_{\mathtt{OT}}$ (as all keys are chosen using fresh and independent randomness), therefore $\Pi_{\mathtt{VOT}}$ securely realizes $\mathcal{F}_{\mathtt{OT}}^{\mathcal{P}}$. Additionally, the above protocol is $\mathcal{P}$-verifiable: indeed, because the entire transcript is broadcasted, the view of any party contains all information needed to check whether or not the transcript is consistent with any given set of inputs and committed randomness. For simplicity, in the following we state the security in the $\{CRS, \hat{\mathcal{F}}_{\text{COM}}\}$-*hybrid model* i.e., where, in addition to the CRS the protocol can make ideal calls to a (one-to-many) commitment functionality $\hat{\mathcal{F}}_{\text{COM}}$ which behaves exactly as $\mathcal{F}_{\text{COM}}$ but allows both public and private opening of the committed value. [12] We point out that all security statements in the lemma are with respect to an adaptive adversary.

Lemma 1. *Assuming $\Pi_{\mathtt{OT}}$ UC securely realizes the two-party 12OT functionality $\mathcal{F}_{OT}$ in the CRS model, the protocol $\Pi_{\mathtt{VOT}}$ (defined above) satisfies the following properties: (security) $\Pi_{\mathtt{VOT}}$ UC securely realizes the multi-party extension $\mathcal{F}_{OT}^{\mathcal{P}}$ of $\mathcal{F}_{OT}$ (defined above) in the $\{CRS, \hat{\mathcal{F}}_{\text{COM}}\}$-hybrid model; ($\mathcal{P}$-verifiability) $\Pi_{\mathtt{VOT}}$ is $\mathcal{P}$-verifiable. Furthermore, $\Pi_{\mathtt{VOT}}$ makes black-box use of (the next-message function of) $\Pi_{\mathtt{OT}}$.*

$\mathcal{P}$-verifiable MPC with (non-identifiable) abort. The next step is to add verifiability to a given adaptively UC secure OT-hybrid MPC protocol $\Pi^{\mathcal{F}_{\mathtt{OT}}}$. Wlog, we assume that $\Pi^{\mathcal{F}_{\mathtt{OT}}}$ only makes calls to $\mathcal{F}_{\mathtt{OT}}$ and to a broadcast channel. (Indeed, $\mathcal{F}_{\mathtt{OT}}$ can be used to also implement secure bilateral communication as follows: to send message x, the sender inputs (x, x) and the receiver input $b = 1$.)

Denote by $\Pi^{\Pi_{\mathtt{VOT}}}$ the version of $\Pi^{\mathcal{F}_{\mathtt{OT}}}$ which starts off by having every party publicly commit to its random tape and has all calls to $\mathcal{F}_{\mathtt{OT}}$ replaced by invocations of protocol $\Pi_{\mathtt{VOT}}$ instantiated with fresh/independent randomness. More precisely, $\Pi^{\Pi_{\mathtt{VOT}}}$ is derived from $\Pi^{\mathcal{F}_{\mathtt{OT}}}$ as follows:

- Initially every party commits to its random tape using one-to-many commitments.
- All calls to $\mathcal{F}_{\mathtt{OT}}$ (including the ones used as above to implement bilateral communication) are replaced by invocations of protocol $\Pi_{\mathtt{VOT}}$. (The random coing do not need to be committed again; the above commitments are used in the invocations of $\Pi_{\mathtt{VOT}}$.)
- For each party p_i a specific part of p_i's random tape is associated with each invocation of $\Pi_{\mathtt{VOT}}$. This part is used only in this invocation and nowhere else in the protocol.

[12] We can use any of the CRS-based commitment protocols [6,7] to instantiate $\hat{\mathcal{F}}_{\text{COM}}$.

The following lemma states the achieved security, where as in Lemma 1 all security statements are with respect to an adaptive adversary. The proof follows from the security of $\Pi^{\mathcal{F}_{\text{OT}}}$ and the security/$\mathcal{P}$-verifiability of Π_{VOT}.

Lemma 2. *Let $\mathcal{F}$ be a UC functionality and $\Pi^{\mathcal{F}_{OT}}$ be a protocol which unconditionally UC securely realizes $\mathcal{F}$ in the $\mathcal{F}_{OT}$-hybrid model with (non-identifiable) abort, and for a protocol Π_{OT} which UC securely realizes $\mathcal{F}_{OT}$ in the CRS model, let Π_{VOT} be the corresponding $\mathcal{P}$-verifiable protocol (as in Lemma 1). Then protocol $\Pi^{\Pi_{\text{VOT}}}$, defined above, satisfies the following properties: (security) $\Pi^{\Pi_{\text{VOT}}}$ UC securely realizes $\mathcal{F}$ with (non-identifiable) abort in the $\{CRS, \hat{\mathcal{F}}_{\text{COM}}\}$-hybrid model; ($\mathcal{P}$-verifiability) Protocol $\Pi^{\Pi_{\text{VOT}}}$ is $\mathcal{P}$-verifiable. Furthermore, $\Pi^{\Pi_{\text{VOT}}}$ makes black-box use of (the next-message function of) Π_{OT}.*

The Setup Compiler. We next describe the protocol $\Pi^{\mathcal{D}}_{\mathsf{Corr}}$ which securely realizes any given sampling functionality $\mathcal{F}^{\mathcal{D}}_{\mathsf{Corr}}$ (for an efficiently computable distribution $\mathcal{D}$), while making black-box use of a UC secure OT-protocol in the CRS model and ideal calls to $\hat{\mathcal{F}}_{\text{COM}}$. The idea is to, first, have every party commits to its random coins and then invoke $\Pi^{\Pi_{\text{VOT}}}$ to securely realize functionality $\mathcal{F}^{\mathcal{D}}_{\mathsf{Corr}}$ using these coins; if the evaluation aborts, then the parties open their committed randomness and use the detector $\mathcal{D}$ to figure out which party cheated. Because the parties have no inputs, opening their randomness does not violate privacy.

Unfortunately, the above over-simplistic protocol is not simulatable. Intuitively, the reason is that $\Pi^{\Pi_{\text{VOT}}}$ might abort after the adversary has seen his outputs of $\mathcal{F}^{\mathcal{D}}_{\mathsf{Corr}}$, in which case the simulator needs to come up with random coins for the simulated honest parties which are consistent with the adversary's view. We resolve this by the following technical trick, which ensures that $\mathcal{S}$ needs to invoke $\mathcal{F}^{\mathcal{D}}_{\mathsf{Corr}}$ only if the computation of $\Pi^{\Pi_{\text{VOT}}}$ was successful: instead of directly computing $\mathcal{F}^{\mathcal{D}}_{\mathsf{Corr}}$, we use $\Pi^{\Pi_{\text{VOT}}}$ to realize the functionality $\langle \mathcal{F}^{\mathcal{D}}_{\mathsf{Corr}} \rangle$ which computes an authenticated (by means of i.t. signatures) n-out-of-n secret sharing of the output of $\mathcal{F}^{\mathcal{D}}_{\mathsf{Corr}}$. This sharing is then reconstructed by having every party announce its share. The authenticity of the output sharing ensures that either the reconstruction will succeed or a party that did not announce a properly signed share will be caught, in which case the protocol identifies this party.

Theorem 5. *Assuming Π_{OT}, Π_{VOT}, and $\Pi^{\Pi_{\text{VOT}}}$ as in Lemma 2, the protocol $\Pi^{\mathcal{D}}_{Corr}$ securely realizes $\mathcal{F}^{\mathcal{D}}_{Corr}$ with identifiable abort in the CRS model while making black-box use of Π_{OT} and ideal calls to the commitment functionality $\hat{\mathcal{F}}_{\text{COM}}$.*

By combining Theorems 4 and 5 with the universal composition theorem, and instantiating $\Pi^{\Pi_{\text{VOT}}}$ with the IPS protocol [32] we obtain the following corollary.

Corollary 3. *There exists a protocol which UC realizes any given functionality with identifiable abort, while making black-box use of a protocol for UC realizing $\mathcal{F}_{OT}$ and a protocol for UC realizing $\hat{\mathcal{F}}_{\text{COM}}$ in the CRS model.*

The Stand-alone model. The proof of Theorem 5 does not use the equivocality of the commitments. Therefore, assuming an adaptive 12OT protocol and

extractable commitments, it can be carried over to the stand-alone setting. Such extractable commitments can be constructed by making a black-box use of a one-way function [38], which in turns can be obtained via a black-box use of OT [27]. Thus, we get the following result for the stand-alone model (see full version for proof).

Lemma 3 (Stand-alone). *There exists a protocol which securely realizes any given functionality with identifiable abort in the plain model making black-box use of an adaptively secure OT protocol in the plain model.*

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 P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pp. 277–296. Springer, 2011.
2. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computations. In *20th ACM STOC*, pp. 1–10. ACM Press, 1988.
3. R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pp. 169–188. Springer, 2011.
4. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
5. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pp. 136–145. IEEE Computer Society Press, 2001.
6. R. Canetti and M. Fischlin. Universally composable commitments. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pp. 19–40. Springer, 2001.
7. R Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pp. 494–503. ACM Press, 2002.
8. R. Canetti and T. Rabin. Universal composition with joint state. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pp. 265–281. Springer, 2003.
9. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *20th ACM STOC*, pp. 11–19. ACM Press, 1988.
10. S. G. Choi, J. Katz, H. Wee, and H.-S. Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global crs. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pp. 73–88. Springer, 2013.
11. Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, Hoeteck Wee. Simple, Black-Box Constructions of Adaptively Secure Protocols. In *6th TCC*, pp. 387-402, 2009.
12. R. Cleve. Limits on the Security of Coin Flips when Half the Processors Are Faulty. In *18th STOC*, pp. 364-369, 1986.
13. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In B. Pfitzmann, editor, *EUROCRYPT 2001*, pp. 280–299, 2001.
14. I. Damgård, M. Keller, E. Larraia, C. Miles, and N. P. Smart. Implementing AES via an actively/covertly secure dishonest-majority MPC protocol. In I. Visconti and R. De Prisco, editors, *SCN 2012*, volume 7485 of *LNCS*, pp. 241–263. Springer, 2012.

15. I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In J. Crampton, S. Jajodia, and K. Mayes, editors, *18th ESORICS*, volume 8134 of *LNCS*, pp. 1–18, Egham, UK, Sep 9–13, 2003. Springer, 2003.
16. I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pp. 643–662. Springer, 2012.
17. T. K. Frederiksen, T. P. Jakobsen, J. B. Nielsen, P. S. Nordholt, and C. Orlandi. Minilego: Efficient secure two-party computation from general assumptions. In T. Johansson and P. Q. Nguyen *EUROCRYPT 2013*, pp. 537–556, 2013.
18. J. A. Garay, P. D. MacKenzie, M. Prabhakaran, and K. Yang. Resource fairness and composability of cryptographic protocols. In S. Halevi and T.Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pp. 404–428. Springer, 2006.
19. O. Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.
20. O. Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.
21. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In A. Aho, editor, *19th ACM STOC*, pp. 218–229. ACM Press, 1987.
22. S. D. Gordon and J. Katz. Partial fairness in secure two-party computation. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pp. 157–176. Springer, 2010.
23. V. Goyal, C.-K. Lee, R. Ostrovsky, and I. Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd FOCS*, pp. 51–60. IEEE Computer Society, 2012.
24. I. Haitner, Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. Black-Box Constructions of Protocols for Secure Computation. *SIAM J. Comput.* 40(2): 225-266, 2011.
25. M. Hirt and U. M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, 2000.
26. M. Hirt, U. M. Maurer, and V. Zikas. MPC vs. SFE: Unconditional and computational security. In J. Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pp. 1–18. Springer, 2008.
27. R Impagliazzo and M. Luby. One-way functions are essential for complexity-based cryptography. In *30th FOCS*, pp. 230–235. IEEE Computer Society Press, 1989.
28. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pp. 145–161. Springer, 2003.
29. Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pp. 483–500. Springer, 2006.
30. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge from secure multiparty computation. In D. S. Johnson and U. Feige, editors, *39th ACM STOC*, pp. 21–30. ACM Press, 2007.
31. Y. Ishai, R. Ostrovsky, and H. Seyalioglu. Identifying cheaters without an honest majority. In R. Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pp. 21–38. Springer, 2012.

32. Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pp. 572–591. Springer, 2008.
33. J. Katz, U. Maurer, B. Tackmann, and V. Zikas. Universally composable synchronous computation. In A. Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pp. 477–498. Springer, 2013. Full version available at Cryptology ePrint Archive, Report 2011/310.
34. M. Keller, P. Scholl, and N. P. Smart. An architecture for practical actively secure MPC with dishonest majority. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *20th ACM CCS*, pp. 549–560, Berlin, Germany, 2013. ACM Press.
35. E. Kushilevitz, Y. Lindell, and T. Rabin. Information-theoretically secure protocols and security under composition. In J. M. Kleinberg, editor, *38th ACM STOC*, pp. 109–118. ACM Press, 2006.
36. J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra. A new approach to practical active-secure two-party computation. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, pp. 681–700, Springer, 2012.
37. S. J. Ong, D. C. Parkes, A. Rosen, and S. P. Vadhan. Fairness with an honest minority and a rational majority. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pp. 36–53. Springer, 2009.
38. R. Pass and H. Wee. Black-box constructions of two-party protocols from one-way functions. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pp. 403–418. Springer, 2009.
39. C. Peikert, V Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pp. 554–571. Springer, 2008.
40. B. Pfitzmann and M. Waidner. Unconditional byzantine agreement for any number of faulty processors. In A. Finkel and M. Jantzen, editors, *STACS 92*, volume 577 of *LNCS*, pp. 337–350. Springer, 1992.
41. M. O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, 1981. Online version available at `http://eprint.iacr.org/2005/187`.
42. T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *21st ACM STOC*, pp. 73–85. ACM Press, 1989.
43. T. Seito, T. Aikawa, J. Shikata, and T. Matsumoto. Information-theoretically secure key-insulated multireceiver authentication codes. In D. J. Bernstein and T. Lange, editors, *AFRICACRYPT 2010*, volume 6055 of *LNCS*, pp. 148–165. Springer, 2010.
44. H. Seyalioglu. *Reducing Trust When Trust is Essential*. PhD thesis, UCLA, 2012.
45. J. Shikata, G. Hanaoka, Y. Zheng, and H Imai. Security notions for unconditionally secure signature schemes. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pp. 434–449. Springer, 2002.
46. C. Swanson and D. R. Stinson. Unconditionally secure signature schemes revisited. In S. Fehr, editor, *ICITS*, volume 6673 of *LNCS*, pp. 100–116. Springer, 2011.
47. A. C. Yao. Protocols for secure computations. In *23rd FOCS*, pp. 160–164. IEEE Computer Society Press, 1982.
48. V. Zikas, S. Hauser, and U. M. Maurer. Realistic failures in secure multi-party computation. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pp. 274–293. Springer, 2009.