Secure Multiparty Computation with Identifiable Abort via Vindicating Release*

Ran Cohen[1], Jack Doerner[2], Yashvanth Kondi[3], and abhi shelat[4]

[1] Reichman University
[2] Brown University, Technion, Reichman University
[3] Silence Laboratories (Deel)
[4] Northeastern University

Abstract. In the dishonest-majority setting, secure multiparty computation (MPC) with *identifiable abort* (IA) guarantees that honest parties can identify and agree upon at least one cheating party if the protocol does not produce an output. Known MPC constructions with IA rely on generic zero-knowledge proofs, adaptively secure oblivious transfer (OT) protocols, or homomorphic primitives, and thus incur a substantial penalty with respect to protocols that abort without identifiability.

We introduce a new, weaker notion of IA called *input-revealing* IA (IRIA), which can be constructed through selective revealing of committed input values—a technique we call *vindicating release*. We show that this weaker form of IA can be achieved with small concrete overheads for many interesting protocols in the literature, including the pre-processing protocols needed for several state-of-the-art MPC protocols.

We next show how to assemble these IRIA components into an MPC protocol for any functionality with *standard* IA. Such a realization differs minimally in terms of cost, techniques, and analysis from the equivalent realization that lacks identifiability, e.g., our total bandwidth overhead incurred is less than 2x, which is an asymptotic improvement over prior work on IA.

On a practical level, we apply our techniques to the problem of threshold ECDSA, and show that the resulting protocol with standard IA is concretely efficient. On a theoretical level, we present a compiler that transforms any secure protocol into one with standard IA assuming only a variant of statically-corruptable ideal OT.

1 Introduction

When a majority of participants in a secure multiparty computation (MPC) protocol behave maliciously, it is not generally possible to guarantee output delivery for honest parties [30]. Consequently, the standard definition of security permits the adversary to halt the protocol after first learning the outputs of all dishonest parties; correctness and privacy, however, are still guaranteed for honest parties.

* The full version [31] is available online: https://eprint.iacr.org/2023/1136

This regime of *security with abort* has been well-studied, beginning with classical feasibility results [88,57,10], followed by many asymptotic improvements [70,42,17], and culminating in concretely efficient protocols [63,86,87].

Unfortunately, many protocols only achieve security with *unanimous abort* (UA), wherein honest parties can detect an attack but not identify a specific cheater. As a result, these protocols are exposed to *denial-of-service* (DoS) attacks, wherein even a single corrupted party can cause the protocol to abort repeatedly. Such DoS attacks are tolerable in some specific use-cases, e.g., in small, closed systems where investigation can be performed out-of-band [52]; however, they *cannot* be tolerated in many large-scale and/or permissionless MPC applications, such as distributed sampling of *structured reference strings* (SRSes) for public consumption, or threshold signing for blockchains. Among the most commonly used SRSes are RSA moduli [15,62,27,28,44,21] and "the powers of τ setup" [13,16,60,75,32]. Many works on the distributed sampling of these objects and on threshold signatures [24,26] aim to prevent DoS attacks by achieving the stronger notion of *identifiable abort* (IA), which guarantees that in the case of an abort, all honest parties agree upon the identity of at least one corrupted party.

The notions of identifiable abort and unanimous abort were both shown feasible by Goldreich, Micali, and Wigderson [57]. The GMW compiler transforms any protocol that is secure against semi-honest adversaries into a protocol that is secure with IA against malicious adversaries. At a high level, the parties commit to their inputs and then run an augmented coin-tossing protocol to sample a committed random tape for each party, after which they run the original protocol over a broadcast channel, but include with each message a zero-knowledge proof that the message has been computed correctly. This approach is simple, but inefficient due to its non-black-box usage of cryptographic primitives.

A line of follow-up work demonstrated that in the dishonest-majority setting, MPC with unanimous abort can withstand computationally unbounded adversaries (i.e., it requires no cryptographic assumptions) if given access to an ideal *oblivious transfer* (OT) functionality [74,70]. Furthermore, the OT functionality need only be accessed *offline* (i.e., before the parties know their inputs); it can therefore be modeled as a pairwise correlation [9]. On the other hand, Ishai, Ostrovsky, and Seyalioglu [68] ruled out information-theoretic MPC with identifiable abort given *any* pairwise correlation. This implies that IA is separated from any ideal oracle that only interfaces with two parties in an offline phase.

Ishai, Ostrovsky, and Zikas [69] overcame this barrier by constructing a complex n-wise correlation that enables information-theoretic MPC with IA. Specifically, they presented a compiler like the one of GMW, which uses information-theoretic signatures for commitments and a distributed version of the IKOS zero-knowledge proof [67] that is information-theoretically secure given their correlation. They also presented a protocol for the distributed sampling of their correlation that makes black-box use of an adaptively secure OT *protocol*[5] and ideal commitments.

Successive works have abandoned the compiler-oriented approach and combined various n-wise correlations with specific online protocols to achieve better efficiency. Baum et al. [7] presented a variant of the BDOZ protocol [14] that uses homomorphic information-theoretic signatures, and a correlation-sampling protocol based upon somewhat homomorphic encryption; this approach avoids zero-knowledge proofs in the online phase. Spini and Fehr [85] and Cunningham et al. [38] adjusted the SPDZ protocol [42] to achieve IA. Baum et al. [8] focused on boolean (rather than arithmetic) circuits and relaxed the problem to allow computational assumptions in the online phase. Their approach is based upon multi-party garbling [10,63], and makes black-box use of a *statically* secure OT protocol, a pseudorandom function (PRF), and additively homomorphic commitments. Their approach also opens up a number of well-known concrete optimizations such as OT extension and free-XOR.

1.1 Identifying Cheaters by Revealing One's Inputs

The results of Ishai et al. [69] or Baum et al. [7] can be interpreted as reducing the problem of achieving IA for general tasks to the problem of achieving IA for *distributed sampling*. As we have argued above, distributed sampling with IA is also important for sampling SRSes. Our work focuses primarily on this problem.

In distributed sampling protocols, the only private inputs of the participants are their random coins; in the case of an abort, these coins can be safely revealed, because neither the outputs nor the coins themselves will ever be used again. This suggests a seemingly simple approach to IA that avoids zero-knowledge statements concerning the next-message function: parties commit to their randomness, and, in the case of an abort, reveal it. Using the opened commitments, honest parties can emulate the protocol and identify (and agree upon) the first player to cheat.

The problem with this approach arises in the security proof: if the adversary causes an abort *after* learning the outputs, then the simulator must produce an *output-consistent* view for each of the honest parties, i.e., a view that is consistent with both the outputs that the adversary has learned and the earlier messages that were sent by honest parties, which were simulated without knowledge of the output.

Ishai et al. [69] addressed the challenge of output consistency by introducing a technical trick to ensure the random coins are never revealed once the output has been learned. Instead of computing the output directly, they first compute a secret sharing of the output that is authenticated with information-theoretic signatures, using a generic MPC protocol in the OT-hybrid model that has security with (non-identifiable) abort against adaptive adversaries. They then reconstruct the output by broadcasting the shares and verifying the signatures.

[5] By *black-box* use of a protocol, we mean black-box access to the protocol's next-message function, but not access to the code or description of the next-message function. This usage was introduced by Ishai et al. [67].

If an abort occurs before the reconstruction, then no output has yet been defined and the simulator can choose any set of random coins. By the correctness of the protocol, they correlate to an output from the appropriate distribution, and the equivocality of the protocol messages already transmitted is ensured by the adaptive security of the generic MPC protocol and the protocol that realizes the OT oracle. An abort occurs during reconstruction (i.e., after the adversary learns the output) *only* if the adversary broadcasts a signature that does not verify, and this condition is identifiable without opening any random coins.

Baum et al. [8] devised a different simulation strategy that only requires *statically* secure OT. Much like Ishai et al., they address the challenge of output consistency by indirectly revealing the output. Their protocol uses homomorphic commitments instead of information-theoretic signatures, and performs the entire computation in both committed and secret-shared form, in parallel. Afterward, they use statistical checks to ensure the equivalence of the committed outputs and the secret-shared outputs, and then open the commitments. Instead of equivocating protocol messages, their simulator produces messages on behalf of the honest parties by running the actual honest parties' code (for the secret-shared components); this prevents direct extraction of the corrupt parties' coins, but the parallel committed computation can be used to extract instead.

Despite these improvements over earlier approaches, neither of these yields an efficient, implementable protocol with IA.

1.2 Our Contributions

In this work we present a new and efficient approach for constructing MPC with identifiable abort that follows the spirit of modern MPC protocols with unanimous abort.

We first introduce a weakened form of identifiable abort that we dub *input-revealing* identifiable abort (IRIA). We then present a novel approach for realizing sampling functionalities with IRIA that is based upon selectively and *asymmetrically* revealing committed inputs in a specific order to vindicate honest parties when an abort occurs. Unlike the prior works described above, our solution to the output consistency problem *does not* rely upon generic zero-knowledge proofs, adaptively secure primitives, homomorphic primitives, or protocol compilers. Instead, we show that under our approach, messages and randomness can be equivocated when (and sometimes *only* when) they must be by exploiting fundamental asymmetries in the functionalities.

To illustrate the value of this idea, we systematically apply it to several well-known building-block protocols from the MPC literature, and use the resulting augmented protocols with IRIA to construct protocols with *standard* IA for generic tasks and for the specific task of threshold ECDSA signing.

Oblivious Transfer. We modify the PVW OT protocol [80] to give the OT sender the ability to decommit both of its inputs to the public. Our protocol achieves an intermediate form of security, between IA and IRIA, that involves revealing only the *sender's* inputs to the adversary when an abort occurs. Thus our protocol realizes Sender-Committed OT with Sender-Input-Revealing IA (SCOT-SIRIA) and Public Verifiability (PV). Relative to the unmodified PVW protocol, our overhead is essentially additive, with the exception that all messages are sent via broadcast. We also give a simple n-wise correlation with a multiplicative depth of 1 from which SCOT (with PV and any flavor of IA) can be trivially realized. Details are given in Section 3.

Furthermore, we demonstrate that the Softspoken OT extension protocol of Roy [82] can be enhanced to make it sender committing and to achieve SIRIA, using only an ideal SCOT-SIRIA oracle, a programmable random oracle, and a broadcast channel. In other words, SCOT-SIRIA (with PV) can be extended in much the same way as standard OT, and with essentially identical concrete cost.[6] Our enhancement is general enough to be applicable to other OT-extension protocols. Details are given in Section 3.4.

VOLE. We modify the vectorized multiplication (i.e., Vector Oblivious Linear Evaluation, or VOLE) protocol of Doerner et al. [46,47,48] by replacing the protocol's OT oracle with a SCOT-SIRIA oracle, and prove that it becomes committing for both participants. Only a few extra instructions are required to decommit. The modified protocol *statistically* realizes Committed VOLE with IRIA (CVOLE-IRIA) and PV in any finite field. This protocol provides an intuitive argument that IRIA is in some sense easier than full adaptive security, because achieving adaptive security for the same protocol would require the simulator to solve an adversarially-influenced instance of an NP-hard problem. Details are given in Section 4.

MPC Preprocessing. We rewrite the preprocessing protocol of MASCOT [73] in terms of an ideal CVOLE-IRIA oracle. This modularizes the MASCOT preprocessing protocol, which simplifies it considerably relative to its original presentation and reveals the essence of the mechanism by which authenticated Beaver triples are generated with an unauthenticated multiplication primitive. This change *alone* is sufficient to statistically realize the SPDZ or BeDOZa preprocessing functionality with IRIA, and with the property that all parties are committed to their outputs. Details are given in Section 5.

General MPC with Standard IA. We show that when the SPDZ preprocessing is generated by our previously realized IRIA functionality, then the SPDZ online protocol can realize *any* sampling functionality with (standard) IA. Altogether, the chain of realizations we have proposed involves few changes, no new assumptions, and little concrete overhead relative to plain MASCOT. Among the sampling functionalities we can realize is the preprocessing sampler for the IOZ compiler, which allows IA to be achieved for *any* functionality, but there is a more direct and more efficient way to attain this goal.

[6] As in standard OT extension, the number of invocations of the SCOT-SIRIA oracle depends only on the security parameter, and the number of invocations of the random oracle depends upon the size of the extension.

To achieve IA directly for *any* functionality, we construct a preprocessing protocol that produces shared multiplication triples with BeDOZa-style [14] statistical MACs, and commits the parties only to the MAC keys. Baum, Melissaris, Rachuri, and Scholl [6] recently observed that if an abort occurs in the online phase of the BeDOZa protocol, then only the MAC keys, and *not* the private inputs, must be revealed in order to identify a cheater. For the task of sampling, this is less efficient than using SPDZ, but for non-sampling tasks it allows us to achieve IA without the far-greater overhead of sampling information-theoretic signatures, as in prior works [69,7]. Unlike the preprocessing protocol of Baum et al. [6], which is tied to the Learning Parity with Noise (LPN) assumption, ours follows statistically from ideal SCOT-SIRIA, and thus a variety of assumptions.

Threshold ECDSA. In Section 7, we describe a simple variation of the three-round threshold ECDSA protocol of Doerner et al. [48] that uses our CVOLE-IRIA primitive to achieve identifiable abort with only a modest increase in bandwidth and computational costs over the original. Unlike in other protocols presented in this work, the threshold ECDSA protocol we present *does not* always abort when some party cheats. This is an essential feature in the context of threshold cryptography, and highlights an advantage in flexibility for our approach to IA. If cheating always implied an abort, and with it the abandonment of all protocol state and refusal of future interaction, then a single cheating party could make it impossible for all other parties (even fully-honest groups) to sign under the same key in the future. Instead, when a party cheats in the signing phase, it causes an identifiable *failure*: the signing protocol halts, a corrupt party is identified, and no signature is produced, but future signatures may still be attempted (even by the corrupt party).

1.3 Interpretation of Results

Since we can realize the sampling functionality that generates the correlated randomness required by the security-enhancing IOZ compiler [69], there are two theory-oriented interpretations of our results:

- Whereas the IOZ compiler as originally presented required an adaptively secure OT *protocol* to sample its correlated randomness, we show the sufficiency of a statically corruptable *ideal* OT variant involving n parties, of which $n-2$ are passive "listeners." This implies that our n-party ideal functionality is complete for identifiable abort.[7] Plain OT has often been considered the fundamental primitive of MPC with unanimous abort [74,70]. We offer SCOT-IRIA[8] as a candidate for the fundamental primitive of MPC with IA.

[7] That is, ideal SCOT-SIRIA yields a statistical realization for any functionality with (plain) IA, in constant rounds.

[8] That is, a variant of SCOT with (double-sided) IRIA, which is strictly weaker than SCOT-SIRIA.

- Our result can be viewed as decomposing the complex n-wise correlated randomness required by the IOZ compiler[9] into multiple copies of a very simple n-wise correlation with a multiplicative depth of 1. In Section 3, we give an informal argument that no correlation complete for MPC with IA can be too much simpler. This serves as evidence that SCOT-IRIA is a reasonable fundamental primitive.

Putting aside the theoretical implications of this work, we argue that the primitives and protocols we introduce are also asymptotically or even concretely efficient. Specifically:

- We argue that our MASCOT variant with IRIA is useful and independently interesting in the context of Beaver-Triples-as-a-Service [84], and that it provides an essential guarantee that is absent from current incarnations of the concept with minimal extra overhead.
- Via closed-form cost analysis in Section 6, we show that our modifications to the MASCOT and Doerner et al.'s VOLE protocols incur a bandwidth cost overhead of roughly 50% and a round count overhead of 3 if an abort does not occur, or roughly 100% and 6 rounds if an abort does occur, relative to the equivalent unmodified protocols that achieve only security with abort.
- Our threshold ECDSA protocol with IA is well within the efficiency envelope of threshold ECDSA protocols that have already been deployed commercially *without* IA. We believe this to be strong evidence of practicality.

1.4 Related Work

Security with identifiable abort was first explicitly recognized by Aumann and Lindell [5] in the context of covert security. It has been used implicitly in a large number of works both before and after it was named, particularly in the domain of MPC with fairness and guaranteed output delivery [58,35,39,34], fair coin tossing [12,61,4,20], $1/p$-security [59,11], and as a stepping stone to stronger security-notions [64,89,66]. Our results can be used immediately to improve upon all of these applications.

Another line of work on identifiable abort has focused on round-efficiency, yielding (optimal) four-round protocols in the plain model [29], as well as two-round protocols in the CRS model with a precise characterization of when and how many times broadcast must occur to achieve specific security guarantees [33,41,43]. Round-efficient protocols with IA are also known for quantum computations [3]. These works focused on theoretical feasibility within particular constraints, and are not suitable for practical implementation. While our techniques yield generic MPC with IA and constant round complexity, we do not focus on optimizing the precise number of rounds, and instead prioritize the simplicity, modularity, and (where concrete performance is concerned) computational efficiency of our construction.

[9] Or, for that matter, any other multiparty correlation for MPC that achieves IA, such as the one proposed by Baum, Orsini, and Scholl [7].

Another line of work has focused on the complexity of the correlated randomness required to achieve generic MPC with identifiable abort in the information-theoretic setting. Ishai, Ostrovsky, and Zikas [69] proved that n-wise correlations suffice, and Ishai, Ostrovsky, and Seyalioglu [68] proved that pairwise correlations do not. Simkin et al. [83] and Brandt et al. [19] showed that for $n-1$ corruptions, a correlation with cardinality $n-1$ is (surprisingly) sufficient, and Brandt [18] subsequently presented tight setup bounds. In our work we achieve security in the presence of $n-1$ corruptions via a correlation of cardinality n; we make no attempt to reduce the cardinality to $n-1$.

In an concurrent and independent work, Baum, Melissaris, Rachuri, and Scholl [6] show how to achieve IA by transforming a special class of sender-receiver protocols, where only the sender has private input. We view their results as complementary to ours. Both our approach and theirs leverage party asymmetry when aborts occur. However, their proof techniques are entirely different from ours. They employ a notion weaker than adaptive security which they call *online extractability* in order to construct a linearly homomorphic commitment scheme with IA. Whereas we make use of standard ideal functionalities that are tweaked to incorporate input revealing, they make black-box use of *protocols* instead.[5] Our techniques are rooted in the well-known notion of Committed OT, and add a similar property of input-committingness to other well-known abstractions, using each abstraction as a basis for instantiating the next. In contrast, their compiler-based technique relies on revealing input tapes, and the properties they require of their commitment scheme seem to emerge from a very specific construction. They give an instantiation only from the LPN assumption and achieve only fully-generic MPC, whereas our approach can be applied widely and instantiable from several assumptions (e.g., DDH, LWE, or DCR+QR). Because their instantiation is PCG-based, it has an asymptotic advantage over ours, whereas ours seems to have better concrete efficiency for small circuits, and for bespoke protocols such as threshold ECDSA that require only a small number of VOLE operations.

2 Preliminaries

Notation. We use $\mathbb{X}$ for an unspecified domain, $\mathbb{G}$ for a group, $\mathbb{F}$ for a field, $\mathbb{Z}$ for the integers, and $\mathbb{N}$ for the natural numbers. We use λ_{c} and λ_{s} to denote the computational and statistical security parameters, respectively. Group operations are expressed additively, and group elements are given capitalized variables. We use $\mathcal{P}_i$ to indicate an actively participating party with index i; in a typical context, there will be a fixed set of active participants denoted $\mathcal{P}_1, \ldots, \mathcal{P}_n$. A party that observes passively but remains silent is denoted $\mathcal{V}$.

Security and Communication Model. We consider a malicious PPT adversary who can statically corrupt any subset of parties. All of our proofs are expressed in the Universal Composition (UC) framework of Canetti [22]. We note that our techniques do not rely on any specific properties of the framework, and can be applied to any security framework that supports synchronous communication and composability.

As per prior works, we consider all messages to be sent over an authenticated broadcast channel, often denoted by $\mathcal{F}_{\mathsf{BC}}$ but here left implicit, and we do not consider any point-to-point communication. Since generic MPC with identifiable abort implies the ability to broadcast [35], we believe this to be reasonable. Some prior works attempt to optimistically minimize the broadcast usage when all participants are honest. For example, techniques have been introduced to minimize the *number* of distinct broadcast invocations in the online phase [7] and the bandwidth complexity of broadcast [8]; both of these techniques apply to our work as well. In general, an adversary can always fraudulently claim not to have received a message that was supposed to have been sent over a point-to-point channel, forcing action to be taken on the broadcast channel; in many (but perhaps not all) cases, this implies that the adversary can force all communication onto the broadcast channel if it so desires.

By convention, we assume that trivial protocol errors are handled appropriately. For example, if a particular party is expected to send a message in a protocol with identifiable abort, but does not do so (or sends a message that fails to type-check), then we assume the other parties abort and identify the silent party as a cheater.

Security with Abort. Under this notion of security, the adversary may cause the protocol to terminate without producing output for the honest parties, and may condition termination on the corrupt parties' output values. Such failures cannot always be attributed to any particular party. This is the most basic notion of security against malicious adversaries. If the parties are guaranteed to agree on whether an abort has occurred, then the abort is *unanimous*; otherwise it is *selective*. Any protocol with selective abort can be modified to achieve unanimous abort via a single broadcast round at the end.

Security with Identifiable Abort (IA). Under this notion of security, the adversary may cause the protocol to terminate without producing output for the honest parties, and may condition termination on the corrupt parties' outputs. To cause an abort, the adversary must identify one corrupt party to the honest parties.

Security with Input-Revealing Identifiable Abort (IRIA). We introduce this notion; it is similar to but strictly weaker than plain identifiable abort. In the case of an abort, the adversary learns the inputs of all honest parties and any random coins sampled by the functionality.

Security with Sender-Input-Revealing Identifiable Abort (SIRIA). This notion is relevant only to functionalities with asymmetric roles for the invoking parties, and lies between plain IA and IRIA. A functionality that has a sender role and a receiver role has SIRIA if the adversary learns only the input of the sender when an abort occurs.

Guaranteed Output Delivery (GOD). This is the strongest notion of security; it is unrealizable for many tasks [30]. Guaranteed output delivery is typically defined in the *stand-alone* model (e.g., Cohen and Lindell [35]) and cannot be captured in the inherently asynchronous UC framework. Katz et al. [72] proposed a means to model synchronous communication within UC, which captures guaranteed termination. When discussing guaranteed output, we implicitly use this model.

Public Verifiability (PV). We model public verifiability as an abstract modifier for other functionalities. The parties interacting with any particular session of an unmodified functionality become the *active participants* in the modified functionality, but there may be additional *observing verifiers* who can register to receive outputs, but cannot otherwise influence the functionality.

Functionality 2.1. $\llbracket \mathcal{F} \rrbracket_{\mathsf{PV}}$. Public Verifiability for $\mathcal{F}$ [32]

The functionality $\llbracket \mathcal{F} \rrbracket_{\mathsf{PV}}$ is identical to the functionality $\mathcal{F}$, except that it interacts with an arbitrary number of additional *observing verification parties* (all of them denoted by $\mathcal{V}$, as distinct from the *actively participating parties* $\mathcal{P}_1$, $\mathcal{P}_2$, etc.). Furthermore, if *all* actively participating parties are corrupt, then $\llbracket \mathcal{F} \rrbracket_{\mathsf{PV}}$ receives its random coins from the adversary $\mathcal{S}$.

Coin Retrieval: Whenever the code of $\mathcal{F}$ requires a random value to be sampled from the domain $\mathbb{X}$, then sample as $\mathcal{F}$ would if at least one of the active participants is honest. If all active participants are corrupt, then send $(\texttt{need-coin}, \mathsf{sid}, \mathbb{X})$ to $\mathcal{S}$, and upon receiving $(\texttt{coin}, \mathsf{sid}, x)$ such that $x \in \mathbb{X}$ in response, continue behaving as $\mathcal{F}$, using x as the random value.

Observer Registration: Upon receiving $(\texttt{observe}, \mathsf{sid})$ from $\mathcal{V}$, remember the identity of $\mathcal{V}$, and if any message with the same sid is sent to *all* active participants in the future, then send it to $\mathcal{V}$ as well.

Basic Functionalities. The constructions in this paper make use of functionalities that are well known from prior works. We use the same one-to-many commitment functionality $\mathcal{F}_{\mathsf{Com}}$ as Cohen et al. [32], which is similar to the one-to-many commitment functionalities given by Canetti and Fischlin [23] and Canetti et al. [25], except that it omits explicit destination parties in favor of allowing passive verifiers to receive commitments when wrapped with $\llbracket \cdot \rrbracket_{\mathsf{PV}}$. We make use of the CRS-sampling functionality $\mathcal{F}_{\mathsf{CRS}}$ introduced by Canetti et al. [25], which samples a random value from the appropriate distribution and outputs it to all parties. We also make use of the correlation-sampling functionality $\mathcal{F}_{\mathsf{Corr}}$ of Ishai et al. [69], which samples a set of correlated random values when invoked, and outputs each of them to a different party. In both cases, we assume the functionalities to have identifiable abort (though their original descriptions do not); i.e., if the adversary wishes to prevent any honest party from receiving output, it must identify one of its corrupt parties to the honest parties. Finally, we use a standard n-party coin-tossing functionality $\mathcal{F}_{\mathsf{CT}}$ with identifiable abort, which is similar to $\mathcal{F}_{\mathsf{CRS}}$ except that by convention it samples uniformly from some domain, instead of from an arbitrary distribution.

3 Sender-Committed OT

Basic 1-of-2 oblivious transfer (OT) [49] is a complete primitive for multiparty computation with (non-identifiable) abort [74,70], but Ishai et al. [68] have shown a separation between any two-party correlation (including OT) and identifiable abort in the information-theoretic setting. We aim to make a minimal enhancement to the OT functionality such that it is complete for IA, and the aforementioned separation tells us that it is *necessary* that this enhancement involve additional parties. The simplest role we can expect additional parties to take is the role of *passive observers*, who only record whether the enhanced functionality aborts and which party was responsible (but do not observe either active party's secrets in the non-aborting case). This is not enough, however, because a dishonest interaction with the functionality might take the form not of an abort, but of an incorrect input (with respect to some *correct* input distribution specified by the protocol that uses the functionality), and as it stands, there is no way for observers to check the inputs for correctness. Again, we seek the simplest enhancement, which in our judgment is to make the functionality *publicly committing* for the sender; i.e., the sender becomes committed when providing their inputs, and both inputs can be decommitted to the passive verifiers at a later time.[10] We refer to our enhancement as Sender-Committed OT or SCOT.

The weakest form of identifiable abort is *input-revealing* for both active participants, and we will prove that this weak form is complete for IA via Theorems 4.2 and 5.2. However, the VOLE construction for arbitrary fields that we discuss in Section 4 requires that the receiver's inputs remain hidden, and so the variant we give formally has only sender-input-revealing IA.

Functionality 3.1. $\mathcal{F}_{\mathsf{SCOT\text{-}SIRIA}}(\mathbb{X})$**: Sender-Committed OT**

This functionality interacts with two active participants, $\mathcal{P}_{\mathsf{A}}$ (Alice) and $\mathcal{P}_{\mathsf{B}}$ (Bob), and with the ideal adversary $\mathcal{S}$. It is parameterized by a domain $\mathbb{X}$.

OT: On receiving $(\texttt{choose}, \mathsf{sid}, \beta)$ from Bob such that $\beta \in \{0,1\}$ and $\mathcal{P}_{\mathsf{B}} \| \mathcal{P}_{\mathsf{A}} \| \mathsf{sid}' = \mathsf{sid}$ for some fresh sid', send $(\texttt{choice-made}, \mathsf{sid})$ to all parties. On receiving $(\texttt{messages}, \mathsf{sid}, \alpha^0, \alpha^1)$ from Alice such that $\alpha^0, \alpha^1 \in \mathbb{X}$, store Alice's message in memory and send $(\texttt{messages-loaded}, \mathsf{sid})$ to all parties. When both Alice and Bob's messages have been received, send $(\texttt{chosen-message}, \mathsf{sid}, \alpha^\beta)$ to Bob and send $(\texttt{ot-done}, \mathsf{sid})$ to all parties.

Opening: On receiving $(\texttt{open}, \mathsf{sid})$ from Alice, if the record $(\texttt{messages}, \mathsf{sid}, \alpha^0, \alpha^1)$ exists in memory, then send it to all parties and ignore all future instructions with the same sid value.

[10] This change also makes the functionality weakly committing for the receiver, which receives the message corresponding to its choice bit, and is bound to that choice bit by its inability to guess the other message, so long as the messages have sufficient entropy. Our functionality is slightly weaker than previously studied notions of double-sided committing OT [36,54] because our functionality only allows the receiver to decommit its choice bit if the sender also decommits.

Abort: On receiving $(\texttt{abort}, \mathsf{sid})$ from $\mathcal{S}$ at any point, such that $\mathcal{P}_\mathsf{B} \| \mathcal{P}_\mathsf{A} \| \mathsf{sid}' = \mathsf{sid}$, if $\mathcal{P}_c$ is corrupt for some $c \in \{\mathsf{A}, \mathsf{B}\}$, then send $(\texttt{messages}, \mathsf{sid}, \alpha^0, \alpha^1)$ to $\mathcal{S}$ if such a record exists in memory,[a] and regardless send $(\texttt{abort}, \mathsf{sid}, \mathcal{P}_c)$ to all parties and ignore all future instructions with the same sid value.

[a] This functionality can be transformed into $\mathcal{F}_{\mathsf{SCOT\text{-}IRIA}}$, with double-sided input-revealing IA, simply by releasing β along with α^0 and α^1 in the case of abort, and it can be transformed into $\mathcal{F}_{\mathsf{SCOT\text{-}IA}}$, with *standard* IA, by releasing none of these values.

In the full version [31] of this paper we explore a simple n-wise correlation, from which our functionality can be realized information-theoretically, and in Section 3.1 we give a direct *computational* realization based on the PVW OT [80] augmented with a simple sigma protocol.

3.1 Direct Computational Realization via PVW

Peikert et al. [80] proposed a composable OT framework (the *PVW framework*, recently honored with the CRYPTO 2023 Test of Time Award) instantiable from the Decisional Diffie-Hellman (DDH), Quadratic Residuosity (QR), or Learning with Errors (LWE) [81] assumptions. We observe that their protocol is *already* committing for the sender, and we need only add an additional phase to the protocol by which the sender can decommit. Prior works [71,53] have added a similar decommitment capability simply by revealing the sender's randomness. However, this approach does not seem to be simulation-secure: for example, in the DDH-based instantiation, simulation of a decommitment for an arbitrary pair of messages seems to require breaking the Discrete-Logarithm (DL) assumption, which invalidates the scheme.

We take a different approach: the sender reveals its inputs, and then proves that they are correct decommitments with respect to the messages already transmitted. Although our protocol achieves UC security, we do *not* require a straight-line-extractable proof of knowledge to do this, because our simulator does not require anything additional to be extracted from the sender. Instead, we combine a simple sigma protocol with a commitment scheme that is binding and equivocable, but *not* extractable, and we show that if a corrupt sender cheats, then there exists a rewinding reduction that breaks the binding property of the commitment scheme. This allows our protocol to achieve composable security while avoiding the compromises typically associated with composable zero-knowledge proofs: specifically, the computation and communication overheads and the programming of the random oracle associated with straight-line extractors [78,51,76].[11] Our approach is general: it can be applied to any instantiation of the PVW

[11] We note that in the UC model, the simulator is forbidden to rewind the environment, which precludes rewinding-based simulation techniques, but reductions run the whole experiment, including the environment, as a subroutine, and thus they can make use of rewinding in the usual way. See, for example, Dodis, Shoup, and Walfish [45].

framework, provided that the underlying *dual-mode encryption* scheme supports a sigma protocol for *proof of correct encryption*, which we define below. In the full version of this paper [31], we also give an explicit construction of our approach based upon the DDH instantiation of PVW, Pedersen commitments, and a variant of the Schnorr protocol.

In the original PVW scheme, all messages correspond to valid inputs, and aborts only occur if parties fail to speak. In our new decommitment phase, a corrupt receiver can cause an abort after the sender's inputs have been revealed. Thus, our approach yields security with sender-input-revealing IA.

3.2 Building Blocks for PVW

Dual-Mode Encryption. Peikert et al. introduced a primitive that they refer to as *dual-mode* encryption, a form of public-key encryption in the CRS model with an added notion of *branches* and two *modes*. Given some public key, messages can be encrypted to any branch. In *messy* mode, some branches are decryptable in the usual sense with a corresponding secret key, whereas some branches are *messy*, meaning that the resulting encryption contains no information about the message. In messy mode, all public keys (including ill-formed ones) contain some messy branches, and messy branches can be efficiently distinguished from decryptable ones if and only if the distinguisher has knowledge of a trapdoor that is embedded in the CRS. In *decryptable* mode, on the other hand, the trapdoor can be used to sample public keys that have *only* decryptable branches. The two modes are efficiently distinguishable if and only if the distinguisher has knowledge of the trapdoor.

More formally, a two-branch dual-mode encryption scheme is a tuple of algorithms $(\mathsf{DMSetup}, \mathsf{DMKeyGen}, \mathsf{DMEnc}, \mathsf{DMDec}, \mathsf{DMTrapKeyGen}, \mathsf{DMFindMessy})$ such that:

1. $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{DMSetup}(1^\lambda, \mu)$ samples a CRS and a trapdoor, given the mode $\mu \in \{0,1\}$, where $\mu = 0$ indicates *messy* mode, and $\mu = 1$ indicates *decryptable* mode. We require that CRSes produced in the two modes are computationally indistinguishable unless the trapdoor is known. That is,
$$\left\{\mathsf{crs} : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{DMSetup}(1^\lambda, 0)\right\} \approx_c \left\{\mathsf{crs} : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{DMSetup}(1^\lambda, 1)\right\}$$
2. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{DMKeyGen}(\mathsf{crs}, \beta)$ samples a public key pk, along with the secret key sk for branch $\beta \in \{0,1\}$.
3. $\tilde{m} \leftarrow \mathsf{DMEnc}(\mathsf{crs}, \mathsf{pk}, b, m)$ emits an encryption $\tilde{m}$ of message $m \in \mathbb{M}$ under branch $b \in \{0,1\}$ of pk. If b is a messy branch of pk, then $\tilde{m}$ contains no information about m. If b is a decryptable branch, then $\tilde{m}$ is semantically secure in the usual way.
4. $m := \mathsf{DMDec}(\mathsf{crs}, \mathsf{sk}, \tilde{m})$ emits a message $m \in \mathbb{M}$ if $\tilde{m}$ is the encryption of m under a decryptable branch of the public key pk corresponding to sk. In other words, we require that for every m in the message space and $(\beta, \mu) \in \{0,1\}^2$
$$\begin{aligned} m = \mathsf{DMDec}(\mathsf{crs}, \mathsf{sk}, \mathsf{DMEnc}(\mathsf{crs}, \mathsf{pk}, \beta, m)) :& \\ (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{DMSetup}(1^\lambda, \mu), (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{DMKeyGen}(\mathsf{crs}, \beta)& \end{aligned}$$

with probability 1 over the coins of the algorithms. This is *perfect correctness.*

5. For descriptions of $\mathsf{DMTrapKeyGen}$ and $\mathsf{DMFindMessy}$ and their properties, we refer the reader to Peikert et al. [80] and the full version of this paper [31].

In addition to the above mentioned properties, we note that ciphertext validity is a public property; i.e., it must be possible to determine from only the CRS, public key, and ciphertext value whether decryption of a ciphertext is possible. This is easily satisfied if all ciphertexts decrypt under all keys.

Proof of Correct Encryption. Under our modification, the sender must be able to prove that a ciphertext is a valid encryption of a particular input, even when the ciphertext is messy. It does this by proving knowledge of the coins used by the encryption algorithm. That is, if we let $\mathsf{DMEnc}(\mathsf{crs}, \mathsf{pk}, b, m; r)$ be a deterministic algorithm that takes the randomness r as input rather than sampling it internally, then the relation

$$\mathcal{R}_{\mathsf{DMEnc}}((\mathsf{crs}, \mathsf{pk}, b, m, \tilde{m}), r) \mapsto \tilde{m} = \mathsf{DMEnc}(\mathsf{crs}, \mathsf{pk}, b, m; r)$$

defines membership in a language for which we must construct a zero-knowledge proof of knowledge of a witness:

$$\mathcal{L}_{\mathsf{DMEnc}} = \{(\mathsf{crs}, \mathsf{pk}, b, m, \tilde{m}) : \exists r \text{ s.t. } \mathcal{R}_{\mathsf{DMEnc}}((\mathsf{crs}, \mathsf{pk}, b, m, \tilde{m}), r) = 1\}$$

Sigma Protocols. A sigma protocol is a three-message protocol for proving knowledge of the witness w for some element x of a language $\mathcal{L}$. It comprises a quartet of algorithms $(\Sigma^1_{\mathcal{L}}, \Sigma^2_{\mathcal{L}}, \Sigma^3_{\mathcal{L}}, \Sigma^{\mathsf{V}}_{\mathcal{L}})$, which respectively generate the three messages of the protocol and verify the proof. For a formal description of these algorithms and their properties, we refer the reader to Damgård [40] and to the full version of this paper [31].

Equivocable Commitments in the CRS Model. Finally, we require a commitment scheme that is much too weak to realize the ideal commitment functionality $\mathcal{F}_{\mathsf{Com}}$: one that is hiding, binding, and equivocable, but not necessarily extractable. An equivocable commitment scheme in the CRS model is a tuple of algorithms $(\mathsf{CSetup}, \mathsf{CCom}, \mathsf{COpen}, \mathsf{CTrapSetup}, \mathsf{CTrapCom}, \mathsf{CEquiv})$ such that:

1. $\mathsf{crs} \leftarrow \mathsf{CSetup}(1^\lambda)$ samples a CRS.
2. $(c, d) \leftarrow \mathsf{CCom}(\mathsf{crs}, m)$ samples a commitment c and an opening d for a message m. We require that the commitment be *hiding*; i.e., that the adversary have negligible advantage in distinguishing a commitment c to m from a commitment c' to m', even given knowledge of crs and choice of m and m'.
3. $\mathsf{COpen}(\mathsf{crs}, c, d)$ is a deterministic algorithm that outputs $\perp$ if d not is a valid opening for c, or m if d is a valid opening for c and c is a commitment to m. We require that the commitment be *binding*; i.e., that it be infeasible for the adversary to find two different valid openings d and d' that cause one commitment c to open to two different messages m and m'.
4. For descriptions of $\mathsf{CTrapSetup}$, $\mathsf{CTrapCom}$, CEquiv, and their properties, we refer the reader to Di Crescenzo et al. [37] and to the full version [31].

3.3 The Modified PVW Scheme

Now we give our SCOT-SIRIA protocol based on PVW OT. We describe it and prove it in terms of general primitives, much as Peikert et al. did, and present a proof of security in the full version [31]. We define a simple functionality $[\![\mathcal{F}_{\mathsf{CRS}}^{\mathsf{DME}}]\!]_{\mathsf{PV}}$ that samples CRSes of the type required by the messy mode of our dual-mode cryptosystem. We also define $[\![\mathcal{F}_{\mathsf{CRS}}^{\mathsf{Com}}]\!]_{\mathsf{PV}}$ that samples CRSes of the type required by our equivocable commitment scheme.

We require one additional primitive, which is an efficiently invertible injective map $\mathsf{Map} : \mathbb{X} \to \mathbb{M}$ from the message domain $\mathbb{X}$ of the protocol to the message domain $\mathbb{M}$ of the dual-mode encryption scheme.

Protocol 3.2. $\pi_{\mathsf{SCOT\text{-}SIRIA\text{-}PV\text{-}PVW}}(\mathbb{X}, \lambda_{\mathsf{c}})$. **Computational SCOT**

This protocol involves two active participants, $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$, who we refer to as Alice and Bob, an a-priori-unknown number of passive verifiers (collectively denoted $\mathcal{V}$), and the ideal functionalities $[\![\mathcal{F}_{\mathsf{CRS}}^{\mathsf{DME}}]\!]_{\mathsf{PV}}$ and $[\![\mathcal{F}_{\mathsf{CRS}}^{\mathsf{Com}}]\!]_{\mathsf{PV}}$. This protocol is parameterized by the sender's message domain $\mathbb{X}$. It makes use of a dual-mode encryption scheme with a sigma protocol for the language of correct encryptions and an equivocable commitment scheme as defined in Section 3.2.

Initialization: On receiving $(\texttt{init}, \mathsf{sid})$ from the environment, $\mathcal{P}_i$ for $i \in \{\mathsf{A}, \mathsf{B}\}$ sends $(\texttt{sample}, \mathsf{sid}\|1)$ to both $[\![\mathcal{F}_{\mathsf{CRS}}^{\mathsf{DME}}]\!]_{\mathsf{PV}}$ and $(\texttt{sample}, \mathsf{sid}\|2)$ to $[\![\mathcal{F}_{\mathsf{CRS}}^{\mathsf{Com}}]\!]_{\mathsf{PV}}$ and receives $(\texttt{crs}, \mathsf{sid}\|1, \mathsf{dmecrs})$ and $(\texttt{crs}, \mathsf{sid}\|2, \mathsf{comcrs})$.

OT:

1. On receiving $(\texttt{choose}, \mathsf{sid}, \beta)$ from the environment, $\mathcal{P}_{\mathsf{B}}$ samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{DMKeyGen}(\mathsf{dmecrs}, \beta)$ and broadcasts $(\texttt{dmepk}, \mathsf{sid}, \mathsf{pk})$ to all parties.
2. On receiving $(\texttt{dmepk}, \mathsf{sid}, \mathsf{pk})$, $\mathcal{P}_{\mathsf{A}}$ outputs $(\texttt{choice-made}, \mathsf{sid})$ to the environment. On also receiving $(\texttt{messages}, \mathsf{sid}, \alpha^0, \alpha^1)$ from the environment, $\mathcal{P}_{\mathsf{A}}$ samples $r^b \leftarrow \{0,1\}^{\lambda_{\mathsf{c}}}$ and computes[a]
$$\tilde{\alpha}^b \leftarrow \mathsf{DMEnc}(\mathsf{dmecrs}, \mathsf{pk}, b, \mathsf{Map}(\alpha^b); r^b)$$
for $b \in \{0,1\}$, and broadcasts $(\texttt{transfer}, \mathsf{sid}, \tilde{\alpha}^0, \tilde{\alpha}^1)$ to all parties.
3. On receiving the $\texttt{transfer}$ message from $\mathcal{P}_{\mathsf{A}}$, all parties output $(\texttt{ot-done}, \mathsf{sid})$ to the environment.
4. On receiving the $\texttt{transfer}$ message from $\mathcal{P}_{\mathsf{A}}$, party $\mathcal{P}_{\mathsf{B}}$ computes
$$\gamma := \mathsf{Map}^{-1}(\mathsf{DMDec}(\mathsf{dmecrs}, \mathsf{sk}, \tilde{\alpha}^\beta))$$
and outputs $(\texttt{chosen-message}, \mathsf{sid}, \gamma)$ to the environment.

Opening:

5. On receiving $(\texttt{open}, \mathsf{sid})$ from the environment, $\mathcal{P}_{\mathsf{A}}$ samples

$$\begin{aligned}(a^b, s^b) &\leftarrow \Sigma^1_{\mathsf{DMEnc}}((\mathsf{dmecrs}, \mathsf{pk}, b, \mathsf{Map}(\alpha^b), \tilde{\alpha}^b), r^b) \qquad \text{for } b \in \{0,1\}\\ (c, d) &\leftarrow \mathsf{CCom}(\mathsf{comcrs}, (a^0, a^1))\end{aligned}$$

   and broadcasts $(\texttt{open-com}, \mathsf{sid}, \alpha^0, \alpha^1, c)$ to all parties.
6. On receiving the $\texttt{open-com}$ message from $\mathcal{P}_{\mathsf{A}}$, party $\mathcal{P}_{\mathsf{B}}$ samples

$$e^b \leftarrow \Sigma^2_{\mathsf{DMEnc}}((\mathsf{dmecrs}, \mathsf{pk}, b, \alpha^b, \tilde{\alpha}^b))$$

   for $b \in \{0,1\}$ and broadcasts $(\texttt{open-chal}, \mathsf{sid}, e^0, e^1)$ to all parties.[b]
7. On receiving the $\texttt{open-chal}$ message from $\mathcal{P}_{\mathsf{B}}$, party $\mathcal{P}_{\mathsf{A}}$ computes

$$z^b \leftarrow \Sigma^3_{\mathsf{DMEnc}}((\mathsf{dmecrs}, \mathsf{pk}, b, \mathsf{Map}(\alpha^b), \tilde{\alpha}^b), r^b, s^b, e^b)$$

   for $b \in \{0,1\}$ and broadcasts $(\texttt{open-resp}, \mathsf{sid}, z^0, z^1, d)$ to all parties.
8. On receiving $(\texttt{open-resp}, \mathsf{sid}, z^0, z^1, d)$ from $\mathcal{P}_{\mathsf{A}}$, all others compute

$$(a^0, a^1) := \mathsf{COpen}(\mathsf{crscom}, c, d)$$

   and output $(\texttt{messages}, \mathsf{sid}, \alpha^0, \alpha^1)$ to the environment if

$$\Sigma^{\mathsf{V}}_{\mathsf{DMEnc}}((\mathsf{dmecrs}, \mathsf{pk}, b, \mathsf{Map}(\alpha^b), \tilde{\alpha}^b), a^b, e^b, z^b) = 1$$

   for $b \in \{0,1\}$. On the other hand, if either of these equations does not hold, then the parties output $(\texttt{abort}, \mathsf{sid}, \mathcal{P}_{\mathsf{A}})$ to the environment.

Verification:

9. If there is an observing verifier $\mathcal{V}$, then upon receiving $(\texttt{observe}, \mathsf{sid})$ from the environment, $\mathcal{V}$ sends $(\texttt{observe}, \mathsf{sid})$ to $[\![\mathcal{F}^{\mathsf{DME}}_{\mathsf{CRS}}]\!]_{\mathsf{PV}}$ and $[\![\mathcal{F}^{\mathsf{Com}}_{\mathsf{CRS}}]\!]_{\mathsf{PV}}$. It then receives the broadcast messages and produces outputs to the environment as described in the forgoing protocol.

[a] We assume without loss of generality that the encryption routine requires only security-parameter many random bits.

[b] In the non-programmable global random-oracle model the parties can instead apply the Fiat-Shamir transform [50]: they can calculate $(e^0, e^1) \leftarrow \mathsf{RO}(\mathsf{sid}, c)$ instead of receiving (e^0, e^1) from $\mathcal{P}_{\mathsf{B}}$. This reduces the **Open** phase of the protocol to a single message.

Theorem 3.3. If $(\mathsf{DMKeyGen}, \mathsf{DMEnc}, \mathsf{DMDec})$ belong to a two-branch dual-mode encryption scheme for the domain $\mathbb{M}$, and $(\mathsf{CCom}, \mathsf{COpen})$ belong to an equivocable commitment scheme, and $(\Sigma^1_{\mathsf{DMEnc}}, \Sigma^2_{\mathsf{DMEnc}}, \Sigma^3_{\mathsf{DMEnc}}, \Sigma^{\mathsf{V}}_{\mathsf{DMEnc}})$ is a sigma protocol for $\mathcal{L}_{\mathsf{DMEnc}}$, and $\mathsf{Map} : \mathbb{X} \to \mathbb{M}$ is an efficiently invertible injective map, then $\pi_{\mathsf{SCOT\text{-}SIRIA\text{-}PV\text{-}PVW}}(\mathbb{X}, \lambda_{\mathsf{c}})$ UC-realizes $[\![\mathcal{F}_{\mathsf{SCOT\text{-}SIRIA}}(\mathbb{X})]\!]_{\mathsf{PV}}$ in the

$(\llbracket \mathcal{F}^{\mathsf{DME}}_{\mathsf{CRS}} \rrbracket_{\mathsf{PV}}, \llbracket \mathcal{F}^{\mathsf{Com}}_{\mathsf{CRS}} \rrbracket_{\mathsf{PV}})$-hybrid model, in the presence of a malicious adversary that statically corrupts any number of passive verifiers and at most one of the active participants $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$.

Proof of the above theorem appears in the full version of this paper [31].

Corollary 3.4. If $\mathsf{GroupGen}(1^{\lambda})$ outputs a distribution of cyclic groups relative to which the decisional Diffie-Hellman problem is hard, and $\mathsf{Map} : \mathbb{X} \to \mathbb{G}$ is an invertible injective map, then there exists a protocol that UC-realizes $\llbracket \mathcal{F}_{\mathsf{SCOT\text{-}SIRIA}}(\mathbb{X}) \rrbracket_{\mathsf{PV}}$ in the $(\llbracket \mathcal{F}^{\mathsf{DME}}_{\mathsf{CRS}} \rrbracket_{\mathsf{PV}}, \llbracket \mathcal{F}^{\mathsf{Com}}_{\mathsf{CRS}} \rrbracket_{\mathsf{PV}})$-hybrid model given $(\mathbb{G}, G, q) = \mathcal{G} \leftarrow \mathsf{GroupGen}(1^{\lambda})$, in the presence of a malicious adversary that statically corrupts any number of passive verifiers and at most one of the active participants $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$.

Proof of the above corollary appears in the full version of this paper [31].

On CRS Reuse. We note that as written, our protocol samples individual CRSes for each session, which is wasteful in practice. This is a notational convenience. The CRSes can be sampled once and safely reused, but the simulator requires the CRS to be programmed differently when Alice is corrupt and when Bob is corrupt, which means that each pair of parties will require two CRSes, one for each direction in which an OT can be performed.

3.4 Extending SCOT-SIRIA

In the full version [31], we describe how to construct Sender-Committed OT *Extension* with SIRIA, starting from SCOT-SIRIA. Since OT extension is only a meaningful improvement over plain OT where concrete efficiency is concerned, we allow ourselves the use of a programmable random oracle (and achieve only computational security). Our protocol is derived from the SoftspokenOT protocol of Roy [82]. Whereas $\pi_{\mathsf{SCOT\text{-}SIRIA\text{-}PV\text{-}PVW}}$ vindicates the sender using reactively-transmitted sigma protocol in the case of an abort, our *extension* protocol performs vindication by specifying that the parties simply release certain intermediate protocol values in a carefully crafted sequence. We also introduce a technique that we call *OT Extension Extension*, in which one instance of OT Extension is used as the base OT for many additional instances. OT extension extension is crucial for attaining the concrete efficiency goals outlined in the rest of this paper, as it allows the OT sender to reactively instantiate a polynomially-large number of OT instances with selective decommitments freely interleaved among them, while performing a number of public key operations proportionate *only* to the security parameter. We arrive at the following corollary for our SCOT extension security theorem:

Corollary 3.5. (Informal)**.** There is a protocol of which a single instance UC-realizes polynomially-many reactively-invoked instances of $\llbracket \mathcal{F}_{\mathsf{SCOT\text{-}SIRIA}} \rrbracket_{\mathsf{PV}}$ with the same sender and receiver, where the total number of public-key operations is independent of the number of `choose` and `open` queries.

4 From SCOT-(S)IRIA to CVOLE-IRIA

In this section, we use the SCOT functionality introduced in Section 3 to statistically UC-realize Committed Vector Oblivious Linear Evaluation with (double-sided) Input-Revealing Identifiable Abort (CVOLE-IRIA) over any finite field.

Functionality 4.1. $\mathcal{F}_{\mathsf{CVOLE\text{-}IRIA}}(\mathbb{F}, \ell)$**: Committed Vector OLE**

This functionality interacts with two active participants, $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$, who we refer to as Alice and Bob, and with the ideal adversary $\mathcal{S}$. It is parameterized by a finite field $\mathbb{F}$ and an integer ℓ.

VOLE: On receiving $(\texttt{correlation}, \mathsf{sid}, b)$ from Bob such that $b \in \mathbb{F}$ and $\mathcal{P}_{\mathsf{B}} \| \mathcal{P}_{\mathsf{A}} \| \mathsf{sid}' = \mathsf{sid}$ for some fresh sid', send $(\texttt{correlation-loaded}, \mathsf{sid})$ to all parties. On receiving $(\texttt{vector}, \mathsf{sid}, \mathbf{a})$ from Alice such that $\mathbf{a} \in \mathbb{F}^{\ell}$, send $(\texttt{vector-loaded}, \mathsf{sid})$ to all parties. When both Alice and Bob's messages have been received:

- If Alice is corrupt, then receive $(\texttt{adv-share}, \mathsf{sid}, \mathbf{c})$ from the adversary and compute $\mathbf{d} \coloneqq \{b \cdot \mathbf{a}_i - \mathbf{c}_i\}_{i \in [\ell]}$.
- If Bob is corrupt, then receive $(\texttt{adv-share}, \mathsf{sid}, \mathbf{d})$ from the adversary and compute $\mathbf{c} \coloneqq \{b \cdot \mathbf{a}_i - \mathbf{d}_i\}_{i \in [\ell]}$.
- If neither active participant is corrupt, then sample $\mathbf{c}, \mathbf{d} \leftarrow \mathbb{F}^{\ell}$ uniformly subject to $b \cdot \mathbf{a}_i = \mathbf{c}_i + \mathbf{d}_i$ for every $i \in [\ell]$.

Finally, store $(\texttt{vole}, \mathsf{sid}, \mathbf{a}, b, \mathbf{c}, \mathbf{d})$ in memory and send $(\texttt{share}, \mathsf{sid}, \mathbf{c})$ to Alice and $(\texttt{share}, \mathsf{sid}, \mathbf{d})$ to Bob and $(\texttt{vole-done}, \mathsf{sid})$ to all parties. If instead of sending $(\texttt{adv-share}, \mathsf{sid}, *)$, the adversary sends $(\texttt{abort}, \mathsf{sid}, \mathcal{P}_c)$ such that $\mathcal{P}_c$ is corrupt, then send $(\texttt{vole}, \mathsf{sid}, \mathbf{a}, b)$ to the adversary and send $(\texttt{abort}, \mathsf{sid}, \mathcal{P}_c)$ to all parties, and ignore all future instructions with the same sid value.

Opening: On receiving $(\texttt{open}, \mathsf{sid})$ from $\mathcal{P}_i$ for some $i \in \{\mathsf{A}, \mathsf{B}\}$, send $(\texttt{open-req}, \mathsf{sid}, \mathcal{P}_i)$ to all parties. On receiving such a message from both Alice and Bob, if a record of the form $(\texttt{vole}, \mathsf{sid}, *, *, *, *)$ exists in memory, then send it to all parties and ignore all future instructions with the same sid value. If an honest active participant sends $(\texttt{open}, \mathsf{sid})$ and the adversary sends $(\texttt{abort-open}, \mathsf{sid}, \mathcal{P}_c)$ such that $\mathcal{P}_c$ is corrupt, then send $(\texttt{vole}, \mathsf{sid}, *, *, *, *)$ to $\mathcal{S}$ if such a record exists in memory, and regardless send $(\texttt{abort}, \mathsf{sid}, \mathcal{P}_c)$ to all parties and ignore all future instructions with the same sid value.

We provide two distinct realizations for the above functionality. The first requires only ideal SCOT-IRIA, and serves to demonstrate that this weakest form of SCOT (with double-sided input revealing) is complete for (standard) IA. We defer a description of the protocol and proof to the full version of this paper [31], but provide the relevant theorem here:

Theorem 4.2. For any $\ell \in \mathbb{N}^+$, there is a protocol that statistically UC-realizes $[\![\mathcal{F}_{\textsf{CVOLE-IRIA}}(\mathbb{Z}_2, \ell)]\!]_{\textsf{PV}}$ against a malicious adversary statically corrupting at most one active participant and any number of passive verifiers in the $[\![\mathcal{F}_{\textsf{SCOT-IRIA}}(\mathbb{Z}_2^{\ell+\lambda_s})]\!]_{\textsf{PV}}$-hybrid model. This protocol requires at most 2 invocations of $[\![\mathcal{F}_{\textsf{SCOT-IRIA}}]\!]_{\textsf{PV}}$ in total.

4.1 CVOLE-IRIA Over Any Finite Field

Our main construction of Committed VOLE supports any finite field and achieves statistical security, but it requires that only the *sender*'s inputs be revealed if the underlying SCOT instances abort (in contrast to the construction referenced by Theorem 4.2, which works even if the receiver's inputs are also revealed). This construction is based upon the two-round OT-multiplication protocols of Doerner et al. [46,47,48], who derived their technique from the semi-honest OT-based multiplication protocol of Gilboa [56]. We will give an informal description of Doerner et al.'s protocol, and then explain how the technique of vindicating release can be applied to the protocol in order to achieve IRIA. We leave a formal protocol description and a security sketch to the full version [31].

Gilboa's protocol leverages the observation that oblivious transfer can be used to compute secret shares of the product of an arbitrary value a (known to Alice) and a single bit β (known to Bob). Alice samples a random output δ and two random OT messages α^0 and α^1 such that $\alpha^0 + \delta = 0$, and $\alpha^1 + \delta = a$. The message that bob chooses will be denoted $\gamma = \alpha^\beta$. If Bob's choice bit is 0, then $\delta + \gamma = 0$, and if he inputs 1, then $\delta + \gamma = a$. If the parties supply a method to linearly decompose a full-width multiplication into a sequence of multiplications by single bits,[12] then Bob can apply this method to his full-width input b to transform it into a vector of choice bits $\boldsymbol{\beta}$, Alice can sample a vector of (independent) masks $\boldsymbol{\delta}$ and two vectors of corresponding messages that use a consistently, and if the decomposition method is inverted on both $\boldsymbol{\delta}$ and Bob's vector of received OT messages $\boldsymbol{\gamma}$, then the result is additive shares of $a \cdot b$.

Next, let us consider malicious security. The sole interaction between the parties in the protocol heretofore described is a sequence of OTs. Bob's input b is represented via the OT choice bits, and if the linear decomposition of his inputs is such that any sequence of bits corresponds to a valid input, then he has no means to cheat. On the other hand, regardless of the decomposition method, Alice's input a must be used *consistently* in all of the OT instances. Doerner et al. [46,47,48] introduce a statistical check that ensures (with overwhelming probability) that the OT messages $\boldsymbol{\gamma}$ actually received by Bob are consistent with a single input. *However*, their check does not validate the *unchosen* messages (i.e., $\boldsymbol{\alpha}_i^{1-\boldsymbol{\beta}_i}$ for each $i \in [|\boldsymbol{\beta}|]$), which means that it introduces a *selective-failure* attack, in which Alice can learn some of Bob's choice bits by cheating in a specific way that corresponds to guessing the values of some subset of them, and observing whether the consistency check passes. In order to mitigate this selective-failure attack, Doerner et al. specify a new decomposition

[12] In the case of Gilboa's original protocol, schoolbook multiplication is used.

method for Bob's input b which guarantees that even if the adversary knows b *and* it correctly guesses statistically-many of Bob's choice bits, his remaining choice bits are statistically close to uniform from the adversary's perspective. This decomposition method resembles a bit-fixing randomness extractor, with analysis based upon the leftover hash lemma [65].

Finally, we come to the questions of input-committingness and identifiable abort: our additions to the Doerner et al. protocols. Recall that given an appropriate decomposition function, Bob's only avenue for cheating is to falsely accuse Alice. In the case that Alice is falsely accused, she can vindicate herself simply by releasing both OT messages in each OT instance, in order to prove that they are consistent with a single input a and with the check messages that she has transmitted. Thus, to achieve vindication for Alice, it is sufficient to replace OT with SCOT. SCOT also allows Alice to decommit her input a voluntarily, but in order to extend the same capability to Bob, we need something more.

The problem we have is subtle. For OT messages with sufficient entropy,[13] OT *already* commits Bob to his choice bits, and he can decommit simply by revealing which message he received. There is an issue with simulation, however: when Alice uses an incorrect message pair for some OT instance in the ideal world, the simulator uniformly chooses the corresponding choice bit for Bob, in order to determine whether she should fail the consistency check. The decomposition method of Doerner et al. ensures that any λ_{s} of Bob's real-world choice bits are statistically indistinguishable from uniform. If Alice cheats in any more than λ_{s} OT instances, the simulator simply aborts on Bob's behalf, because in the real world she evades the check with statistically negligible probability. However, if Alice cheats in fewer than λ_{s} OT instances, an abort may not occur, and yet Bob will need to decommit his input. The simulator has already chosen choice bits for Bob for the OT instances corresponding to Alice's cheats, and now it must find a decomposition of his input that is *consistent* with these choice bits. When Doerner et al.'s decomposition method is used, this implies solving an adversarially-influenced instance of the subset sum problem.[14]

We avoid this difficulty by *gradually* releasing protocol values so that Bob decommits his input (and the simulator correspondingly finds a decomposition of his input) *only* if it is guaranteed that Alice has not cheated. If the inputs must be decommitted (or an abort occurs), then the vindication process begins with Bob committing the OT messages $\boldsymbol{\gamma}$ that he received using $\mathcal{F}_{\mathsf{Com}}$. Alice then instructs $\mathcal{F}_{\mathsf{SCOT\text{-}SIRIA}}$ to release her OT messages for all OT instances. Bob (and any external verifiers) verify that Alice has not cheated, and if she has not, then Bob decommits $\boldsymbol{\gamma}$.[15] The simulator is only compelled to sample a full set

[13] Due to Alice's per-instance masks $\boldsymbol{\delta}_i$ for $i \in [|\boldsymbol{\beta}|]$, her OT messages indeed have sufficient entropy.

[14] The subset-sum problem is NP-Complete in general, and we do not know of any strategy under which it is efficiently solvable in the specific parameter regime that the adversary induces.

[15] If Alice was accused of cheating by Bob, and she is vindicated, then the result of this process is an abort in which Bob is identified as corrupt.

of choice corresponding to a specific b in the case that Alice behaved honestly, it can do so because the randomness extractor underlying the decomposition function is *explainable*[16] when no bits are fixed in advance. We note that the success of the above strategy illustrates the intuitive notion that identifiable abort is *easier* than security against adaptive corruptions. To our knowledge, the VOLE protocol we have presented is *not* adaptively secure, because an adversary corrupting Alice could cheat in such a way that when it later corrupts Bob, the simulator is forced to solve an adversarial instance of subset sum.

In the full version of this work [31], we specify our protocol and sketch a proof of the following theorem. We note that our protocol can only be constructed from SCOT-SIRIA, and not from SCOT with double-sided IRIA, due to the same simulation difficulty we have described above.

Theorem 4.3. For any finite field $\mathbb{F}$ and any $\ell \in \mathbb{N}^+$, $\pi_{\mathsf{CVOLE\text{-}IRIA\text{-}PV}}(\mathbb{F}, \ell, \lambda_{\mathsf{s}})$ statistically UC-realizes $[\![\mathcal{F}_{\mathsf{CVOLE\text{-}IRIA}}(\mathbb{F}, \ell)]\!]_{\mathsf{PV}}$ against a malicious adversary statically corrupting at most one active participant and any number of passive verifiers in the $[\![\mathcal{F}_{\mathsf{SCOT\text{-}SIRIA}}]\!]_{\mathsf{PV}}$-hybrid model.

4.2 Single-Sided CVOLE

A simple adjustment of $\mathcal{F}_{\mathsf{CVOLE\text{-}IRIA}}$ gives us a one-sided variant wherein only Alice's inputs and outputs are revealed in the case that an abort occurs, or the **Opening** phase is invoked, and only Alice need participate in order to trigger an **Opening**. This is in some sense analogous to the functionality we specify for OT in Section 3, and it will be useful for constructing the protocols in Sections 5 and 7. Since Alice (who holds the vector) in our VOLE protocol is the OT *sender*, we refer to this functionality as SCVOLE-IRIA:

Functionality 4.4. $\mathcal{F}_{\mathsf{SCVOLE\text{-}IRIA}}(\mathbb{F}, \ell)$**: Sender-Committed VOLE**

This functionality adds the following phase to $\mathcal{F}_{\mathsf{CVOLE\text{-}IRIA}}(n, \mathbb{F})$:

Sender Opening: On receiving $(\texttt{open-sender}, \mathsf{sid})$ from $\mathcal{P}_{\mathsf{A}}$, if a record of the form $(\texttt{vole}, \mathsf{sid}, \mathbf{a}, *, \mathbf{c}, *)$ exists in memory, then send $(\texttt{alice-state}, \mathsf{sid}, \mathbf{a}, \mathbf{c})$ to all parties and ignore all future instructions with the same sid value. Note that this instruction cannot abort.

It is possible to modify the functionality further, such that the *receiver's* input (i.e., Bob's input) is vectorized, rather than Alice's. Per tradition, reversing the roles in an asymmetric protocol results in reversing the name:

Functionality 4.5. $\mathcal{F}_{\mathsf{SCELOV\text{-}IRIA}}(\mathbb{F}, \ell)$**: Sender-Committed ELOV**

This functionality is the same as $\mathcal{F}_{\mathsf{SCVOLE\text{-}SIRIA}}(n, \mathbb{F})$, except that in the **VOLE** phase, Bob sends $(\texttt{vector}, \mathsf{sid}, \mathbf{b})$ such that $\mathbf{b} \in \mathbb{F}^{\ell}$ and Alice sends

[16] Explainable randomness extractors were introduced by Abram et al. [1].

$(\texttt{correlation}, \mathsf{sid}, a)$ such that $a \in \mathbb{F}$, and the constraint on the outputs is $\mathbf{b}_i \cdot a = \mathbf{c}_i + \mathbf{d}_i$ for every $i \in [\ell]$.

Both of these modified functionalities can be realized with simple adjustments to our protocol, which we discuss in the full version of this paper [31]. We also define (and realize) variations with SIRIA instead of IRIA.

5 From CVOLE-IRIA to Generic MPC

We now use the Committed Vector OLE functionalities introduced in Section 4 to construct generic MPC with plain IA via purely information-theoretic techniques. Our first pathway leverages an observation made by Baum et al. [6] in a concurrent work to achieve MPC with IA directly; our second pathway, the details of which are deferred to the full version of this work [31], involves sampling the preprocessing for *classic* information-theoretic MPC-IA online phases. The second pathway allows us to instantiate the security-enhancing IOZ compiler [69] with somewhat weaker prerequisites than were previously known, and thereby add IA to any semi-honest protocol.

5.1 Direct MPC-IA via BeDOZa and Vindicating Release

The classic BeDOZa MPC protocol [14] allows a set of parties to compute any circuit comprising addition and multiplication gates over a field $\mathbb{F}_q$. The parties manipulate data in the form of additive secret shares over $\mathbb{F}_q$. Sharings for each wire in the circuit are generated in a *preprocessing* phase, and manipulated information-theoretically in an *online* phase once the inputs become available. Honest behavior is guaranteed in the online phase using pairwise linear message authentication codes (MACs) on every share of every wire. The pairwise and linear nature of these MACs is the key element that allows us to apply the technique of vindicating release. During the online phase of the BeDOZa protocol, every value transmitted from $\mathcal{P}_i$ to $\mathcal{P}_j$ is accompanied by a tag, and $\mathcal{P}_j$ can use the circuit topology to derive a MAC to verify the tag against the transmitted value under $\mathcal{P}_j$'s secret key. If the verification fails in classic BeDOZa, then $\mathcal{P}_j$ aborts, but there is no way for bystander parties to determine whether $\mathcal{P}_j$'s abort was truly the result of a failed validation, or whether $\mathcal{P}_j$ aborted maliciously. We propose that $\mathcal{P}_j$ can vindicate itself simply by releasing its key and the relevant MAC to the public, after which the other parties can verify the MAC and determine definitively which of the two parties caused the abort to occur.

The observation that public release of keys and MACs allows aborts to be attributed in the BeDOZa protocol was first made by Baum et al. in a concurrent work [6], although they did not have the terminology of *vindicating release* to describe this idea. They also did not take a modular approach; there is no clean separation in their work between the online and offline phases, and their preprocessing is computed using a monolithic protocol tied specifically to the

Learning Parity with Noise (LPN) assumption. We propose a modular formulation. First, we define a BeDOZa-IA preprocessing functionality (Functionality 51) that can *reactively* and *selectively* unveil the MACs and keys necessary for attributing aborts. Second, we introduce a much simpler preprocessing protocol that structurally resembles the well-known MASCOT protocol [73]. This preprocessing protocol is information-theoretically secure assuming ideal CV-OLE and SCELOV functionalities (as introduced in Section 4), and like the other constructions in the work, it achieves IA via vindicating release.

For formal details of the BeDOZa online phase, we refer the reader to Bendlin et al. [14] and Baum et al. [6]. The (modified) BeDOZa online phase UC-realizes a standard notion of generic MPC with IA, as given by Ishai et al. [69]. Before we describe our preprocessing functionality, we must give some notation.

Notation. Our functionality represents elements of the BeDOZa preprocessing using a specific convention. The party to whom a value belongs is always indexed first, so, for example, $\mathbf{x}_{i,*}$ is a vector of $\mathcal{P}_i$'s shares. There are v Beaver triples; each triple contains the input wire sharings $\mathbf{x}_{*,k}$ and $\mathbf{y}_{*,k}$ and the output (product) wire sharing $\mathbf{z}_{*,k}$. There are also $\mathbf{u}_j$ input wires allocated for $\mathcal{P}_j$; the k^{th} input wire for $\mathcal{P}_j$ is shared as $w_{*,j,k}$ and the mask for this wire is $\mathbf{m}_{j,k}$. Each party has a MAC on every share of every wire. $\mathcal{P}_i$ uses the key $\boldsymbol{\delta}_{i,j}$ to MAC the wires of $\mathcal{P}_j$, and $\mathcal{P}_i$'s MAC on $\mathbf{x}_{j,k}$ (i.e., $\mathcal{P}_j$'s share of the k^{th} Beaver triple's left input wire) is a vector $\mathring{\mathbf{x}}_{i,j,k,*}$ of length ρ such that $\rho \cdot |q| \geq \lambda_{\mathsf{s}}$, while $\mathcal{P}_j$'s corresponding tag is $\tilde{\mathbf{x}}_{j,i,k,*}$.

Functionality 5.1. $\mathcal{F}_{\mathsf{BeDOZaPrep\text{-}IA}}(n, \mathbb{F}_q, \lambda_{\mathsf{s}})$**: BeDOZa Preprocessing**

This functionality interacts with n active participants, $\mathcal{P}_1, \ldots, \mathcal{P}_n$. In addition to the party count, it is parameterized by a finite field $\mathbb{F}_q$ of size q over which the correlated randomness is generated, and a statistical parameter λ_{s}. For convenience, we define $\rho := \lceil \lambda_{\mathsf{s}}/|q| \rceil$ to be the number of $\mathbb{F}_q$ elements required for a λ_{s}-bit MAC.

BeDOZa Preprocessing: On receiving $(\texttt{prep}, \mathsf{sid}, \mathbf{u}, v)$ from some $\mathcal{P}_i$ such that $\mathcal{P}_1 \| \ldots \| \mathcal{P}_n \| \mathsf{sid}' = \mathsf{sid}$ for some fresh sid' and $|\mathbf{u}| = n$, send $(\texttt{prep-req}, \mathsf{sid}, \mathbf{u}, v, \mathcal{P}_i)$ to all parties. On receiving $(\texttt{prep}, \mathsf{sid}, \mathbf{u}, v)$ from every $\mathcal{P}_i$ for $i \in [n]$, wait for $\mathcal{S}$ to send $(\texttt{proceed}, \mathsf{sid})$ or $(\texttt{abort}, \mathsf{sid}, \mathcal{P}_c)$ such that $c \in [n]$ and $\mathcal{P}_c$ is corrupt. In the latter case, send $(\texttt{abort}, \mathsf{sid}, \mathcal{P}_c)$ to all parties and ignore future messages with the same sid. In the former case:

1. For every $i \in [n]$ such that $\mathcal{P}_i$ is corrupt, wait for $\mathcal{S}$ to send $(\texttt{adv-shares}, \mathsf{sid}, i, \boldsymbol{\delta}_{i,[\rho]}, \mathbf{m}_{i,[\mathbf{u}_i]}, \{\mathbf{w}_{i,j,[\mathbf{u}_j]}\}_{j \in [n]}, \mathbf{x}_{i,[v]}, \mathbf{y}_{i,[v]}, \mathbf{z}_{i,[v]})$.
2. For every $i \in [n]$ such that $\mathcal{P}_i$ is honest, sample

$$\boldsymbol{\delta}_{i,[\rho]} \leftarrow \mathbb{F}_q^{\rho} \qquad \mathbf{m}_{i,[\mathbf{u}_i]} \leftarrow \mathbb{F}_q^{\mathbf{u}_i} \qquad \mathbf{w}_{i,j,[\mathbf{u}_j]} \leftarrow \mathbb{F}_q^{\mathbf{u}_j} \quad \text{for } j \in [n]$$

$$\mathbf{x}_{i,[v]} \leftarrow \mathbb{F}_q^{v} \qquad \mathbf{y}_{i,[v]} \leftarrow \mathbb{F}_q^{v} \qquad \mathbf{z}_{i,[v]} \leftarrow \mathbb{F}_q^{v}$$

subject for every $k \in [v]$ to

$$\sum_{j\in[n]} \mathbf{z}_{j,k} = \sum_{j\in[n]} \mathbf{x}_{j,k} \cdot \sum_{j\in[n]} \mathbf{y}_{j,k}$$

and subject to

$$\mathbf{m} = \left\{ \left\{ \sum_{i\in[n]} \mathbf{w}_{i,j,k} \right\}_{k\in[\mathbf{u}_j]} \right\}_{j\in[n]}$$

3. For every $i \in [n]$ and $h \in [n] \setminus \{i\}$, sample

$$\mathring{\mathbf{x}}_{i,h,[v],[\rho]}, \mathring{\mathbf{y}}_{i,h,[v],[\rho]}, \mathring{\mathbf{z}}_{i,h,[v],[\rho]} \leftarrow \mathbb{F}_q^{v\times\rho}$$

$$\tilde{\mathbf{x}}_{i,h,[v],[\rho]}, \tilde{\mathbf{y}}_{i,h,[v],[\rho]}, \tilde{\mathbf{z}}_{i,h,[v],[\rho]} \leftarrow \mathbb{F}_q^{v\times\rho}$$

$$\mathring{\mathbf{w}}_{i,h,j,[\mathbf{u}_j],[\rho]}, \tilde{\mathbf{w}}_{i,h,j,[\mathbf{u}_j],[\rho]} \leftarrow \mathbb{F}_q^{\mathbf{u}_j\times\rho} \quad \text{for } j \in [n]$$

subject for every $k \in [v]$ and $l \in [\rho]$ to

$$\begin{aligned} \mathring{\mathbf{x}}_{i,h,k,l} + \tilde{\mathbf{x}}_{h,i,k,l} &= \boldsymbol{\delta}_{i,l} \cdot \mathbf{x}_{h,k} \\ \mathring{\mathbf{y}}_{i,h,k,l} + \tilde{\mathbf{y}}_{h,i,k,l} &= \boldsymbol{\delta}_{i,l} \cdot \mathbf{y}_{h,k} \\ \mathring{\mathbf{z}}_{i,h,k,l} + \tilde{\mathbf{z}}_{h,i,k,l} &= \boldsymbol{\delta}_{i,l} \cdot \mathbf{z}_{h,k} \end{aligned}$$

and subject for every $j \in [n]$ and $k \in [\mathbf{u}_j]$ and $l \in [\rho]$ to

$$\mathring{\mathbf{w}}_{i,h,j,k,l} + \tilde{\mathbf{w}}_{h,i,j,k,l} = \boldsymbol{\delta}_{i,l} \cdot \mathbf{w}_{h,j,k}$$

4. Output

$$\begin{pmatrix} \mathtt{prep}, \mathsf{sid}, \boldsymbol{\delta}_{i,*}, \mathbf{m}_{i,*}, \mathbf{w}_{i,*,*}, \mathring{\mathbf{w}}_{i,*,*,*,*}, \tilde{\mathbf{w}}_{i,*,*,*,*}, \\ \qquad \mathbf{x}_{i,*}, \mathring{\mathbf{x}}_{i,*,*,*}, \tilde{\mathbf{x}}_{i,*,*,*},, \mathbf{y}_{i,*}, \mathring{\mathbf{y}}_{i,*,*,*}, \tilde{\mathbf{y}}_{i,*,*,*}, \mathbf{z}_{i,*}, \mathring{\mathbf{z}}_{i,*,*,*}, \tilde{\mathbf{z}}_{i,*,*,*} \end{pmatrix}$$

to $\mathcal{P}_i$ for every $i \in [n]$ and store $(\mathtt{prep}, \mathsf{sid}, \boldsymbol{\delta}, \mathbf{m}, \mathbf{w}, \mathring{\mathbf{w}}, \tilde{\mathbf{w}}, \mathbf{x}, \mathring{\mathbf{x}}, \tilde{\mathbf{x}}, \mathbf{y}, \mathring{\mathbf{y}}, \tilde{\mathbf{y}}, \mathbf{z}, \mathring{\mathbf{z}}, \tilde{\mathbf{z}})$ in memory.

Opening: On receiving $(\mathtt{open}, \mathsf{sid}, \mathcal{P}_j)$ from some $\mathcal{P}_i$ such that $j \in [n]$, find the record $(\mathtt{prep}, \mathsf{sid}, \boldsymbol{\delta}, \mathbf{m}, \mathbf{w}, \mathring{\mathbf{w}}, \tilde{\mathbf{w}}, \mathbf{x}, \mathring{\mathbf{x}}, \tilde{\mathbf{x}}, \mathbf{y}, \mathring{\mathbf{y}}, \tilde{\mathbf{y}}, \mathbf{z}, \mathring{\mathbf{z}}, \tilde{\mathbf{z}})$ and send $(\mathtt{macs}, \mathsf{sid}, \mathcal{P}_i, \mathcal{P}_j, \boldsymbol{\delta}_{i,*}, \mathring{\mathbf{w}}_{i,j,*,*,*}, \mathring{\mathbf{x}}_{i,j,*,*}, \mathring{\mathbf{y}}_{i,j,*,*}, \mathring{\mathbf{z}}_{i,j,*,*})$ to all parties.

In order to realize the above functionality, we adjust the MASCOT protocol [73]. We defer a formal description of the protocol and a proof sketch to the full version of this paper [31]. Here we give a high-level overview of our modifications and an intuitive view of their security.

Before we describe how IA is achieved, we make two observations about the structure of the original MASCOT protocol. First, the original protocol generates the preprocessing for the *SPDZ* online phase [42], rather than BeDOZa. In SPDZ, the parties hold shares of a single key, and shares of MAC on each wire's logical value; in contrast, in BeDOZa, each party holds an individual MAC on each share of each wire under its own key. The total size of the SPDZ preprocessing is therefore a factor of n smaller than the BeDOZa preprocessing. Since the MACs are linear in both cases, though, the equations that define the preprocessing differ mainly in how the intermediate values are summed, which means that almost any mechanism to generate the preprocessing of one can easily be adapted to generate the preprocessing of the other. We can only achieve efficient generic MPC with IA via BeDOZa, but a variant that generates SPDZ preprocessing remains interesting because it can provide an enhanced security guarantee in the Triples-as-a-Service Paradigm [84]. In this context, one set of parties (i.e., service providers) generates the preprocessing that is consumed by another set of parties, who have the inputs. Our functionality allows malicious providers to be identified, even if the online MPC protocol does not achieve IA.

Second, the original MASCOT protocol is monolithic, and it has a monolithic proof. Logically, the protocol can separated into two components: a set of pairwise unauthenticated OLE and VOLE operations with malicious security, and then an information-theoretically secure means to combine the unauthenticated VOLE outputs into a Beaver triple. We separate the protocol in this way, and view this modularization of the MASCOT protocol as a minor contribution.

Achieving IA for the modular MASCOT protocol is easy. We replace the OLE and VOLE invocations with calls to the CVOLE-IRIA functionality. If an abort occurs during the protocol, the parties decommit their inputs and check the transcript for inconsistencies. Because the protocol is information-theoretic, the simulator can easily compute an input-consistent value to decommit on behalf of the honest parties when an abort occurs. The protocol realizes an IA functionality because this is equivalent to IRIA in the context of sampling.

Finally, we need the ability to decommit the MACs and MAC keys publicly in order to perform vindicating release in the main BeDOZa-IA protocol. To accomplish this, we use calls to SCELOV instead of CVOLE in places where the MAC keys are input as correlations. This allows exactly the sort of retroactive decommitment that we need.

In the full version [31], we give a formal protocol specification and prove:

Theorem 5.2. *For any finite field $\mathbb{F}$ and $n, \lambda_{\mathsf{s}} \in \mathbb{N}^+$, there exists a protocol that statistically UC-realizes $[\![\mathcal{F}_{\mathsf{BeDOZaPrep\text{-}IA}}(n, \mathbb{F}, \lambda_{\mathsf{s}})]\!]_{\mathsf{PV}}$ against a malicious adversary statically corrupting up to $n-1$ active parties in the $([\![\mathcal{F}_{\mathsf{CVOLE\text{-}IRIA}}]\!]_{\mathsf{PV}}, [\![\mathcal{F}_{\mathsf{Com}}]\!]_{\mathsf{PV}}, [\![\mathcal{F}_{\mathsf{CT}}]\!]_{\mathsf{PV}})$-hybrid model.*

Next, combining $[\![\mathcal{F}_{\mathsf{BeDOZaPrep\text{-}IA}}]\!]_{\mathsf{PV}}$ with the BeDOZa-IA online phase yields:

Corollary 5.3. *For any n-party ideal functionality $[\![\mathcal{F}_{\mathsf{IA}}]\!]_{\mathsf{PV}}$ that has publicly verifiable identifiable abort,[17] there exists a protocol that statistically UC-realizes $[\![\mathcal{F}_{\mathsf{IA}}]\!]_{\mathsf{PV}}$ against a malicious adversary statically corrupting up to $n-1$ active participants in the $[\![\mathcal{F}_{\mathsf{SCOT\text{-}SIRIA}}]\!]_{\mathsf{PV}}$-hybrid model.*

[17] The public verifiability aspect of this functionality is optional.

Since we can realize any functionality with IA, we can realize the preprocessing functionality for the IOZ compiler [69], which implies:

Corollary 5.4. Let π_{SH} be a $\mathcal{F}^{\mathcal{D}}_{\mathsf{Corr}}$-hybrid protocol (for an efficiently computable distribution $\mathcal{D}$) that information-theoretically UC-realizes a functionality $\mathcal{F}$ in the presence of a semi-honest adversary statically corrupting $n-1$ participants. There exists a compiler to turn π_{SH} into an $[\![\mathcal{F}_{\mathsf{SCOT\text{-}SIRIA}}]\!]_{\mathsf{PV}}$-hybrid protocol that information-theoretically UC-realizes $\mathcal{F}$ with publicly verifiable identifiable abort in the presence of a malicious adversary statically corrupting $n-1$ participants. Moreover, this compiler is asymptotically round-preserving.

6 Cost Analysis

Our protocols for IA are the first ones, as far as we can tell, that are simple enough that we can estimate their costs. In the full version, we perform a closed-form worse-case cost analysis of the protocols we have introduced up to this point. Since our approach in this paper involves parties taking active steps to vindicate their behavior in the case of an abort, we give two sets of cost equations: one for the eventuality of an abort, and one for the eventuality of a successful protocol completion. We count costs separately for each role within a protocol and each protocol phase. Our counts include only data payloads and ignore bookkeeping values such as session IDs, which are implementation-dependent.

In Figure 1, we apply our cost model to our proposed version of MASCOT that provides identifiable abort as described in Theorem 5.2. We divide the protocol costs into two parts. The first cost corresponds to the *initialization* of primitives like CVOLE. This cost needs only to be paid once, if no abort occurs. The second cost corresponds to the *preparation* of triples etc., which can be done repeatedly until an abort occurs, given a single initialization. The figure depicts communication costs for both phases (and the worst-case abort) in order to evaluate a circuit with $u = 10$ inputs and $v = 10^4$ wires for 128-bit security parameters. The model demonstrates that our concrete overhead for achieving IA is a constant factor < 2 over the base protocol.[18]

7 Threshold ECDSA with Identifiable Abort

In prior sections, we explain how our techniques can achieve generic MPC. In this section, we show how our techniques can also be applied *directly* to bespoke protocols such as recent multiparty ECDSA signing protocols in order to augment them with identifiable abort. Designing multiparty signing protocols for ECDSA is nontrivial due to its nonlinear nature: a signature on message m under public key $\mathsf{pk} = \mathsf{sk} \cdot G$ is of the form (R, s), where $R = r \cdot G$ is a signing nonce, and $s = (\mathsf{SHA2}(m) + r^{\mathsf{x}} \cdot \mathsf{sk})/r$ is a scalar (r^{x} is the x-coordinate of R).

[18] We note, however, that the basic MASCOT protocol uses point-to-point channels, whereas our protocol requires all data to be transmitted over a broadcast channel.

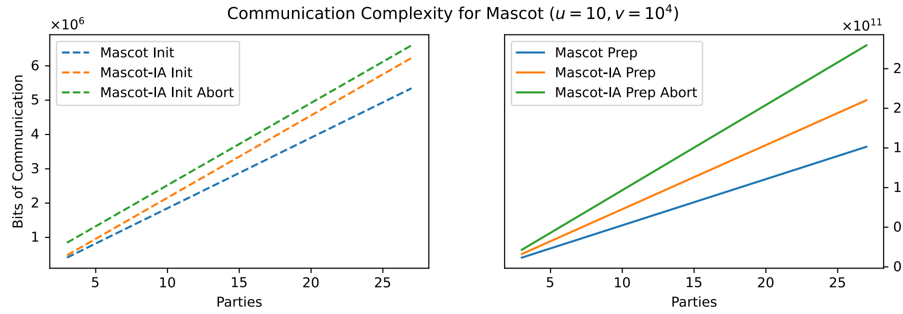

Fig. 1: Concrete communication costs of the initialization and triple preparation phases of the basic version of MASCOT versus our proposed identifiable-abort version. $\lambda_c = 128, \lambda_s = 80$ using a DDH-based OT.

The only existing multiparty ECDSA schemes that achieve identifiable abort are those of Canetti et al. [24] and Castagnos et al. [26], which are based on Paillier encryption and class groups, respectively. They both follow a conceptually simple GMW-style template: they start with a semi-honest protocol that uses homomorphic encryption to carry out the nonlinear operations, and augment it with carefully crafted zero-knowledge proofs to guarantee that all operations are performed honestly. While relatively round-efficient, their approaches are non-black-box in homomorphic encryption and incur considerable overhead; in particular they must prove that ciphertexts are related to one another and to elliptic curve points in various ways, and Canetti et al. must also prove correspondence with plaintexts lying in specific ranges.

In contrast, the protocol that we present in this section makes black-box use of the $[\![\mathcal{F}_{\mathsf{SCVOLE\text{-}SIRIA}}(\mathbb{Z}_q, 2)]\!]_{\mathsf{PV}}$ functionality, which was discussed in Section 4.2 (along with protocol that realizes it), and it completely avoids expensive zero-knowledge proofs over ciphertexts or groups of unknown order. We do make use of zero-knowledge proofs for simple statements that are black-box in the ECDSA signing curve, but these proofs are invoked *only* when an abort actually occurs, and they are instantiable with lightweight, Schnorr-like sigma protocols.

An Intuitive Overview. Our approach is to tweak the protocol of Doerner et al. [48] in two ways: First, we rewrite it to generate a random correlation that is later derandomized to incorporate the signing key and nonce.[19] Using the decommitment interface of $[\![\mathcal{F}_{\mathsf{SCVOLE\text{-}SIRIA}}]\!]_{\mathsf{PV}}$, this correlation can be (partially) opened to the public if it has not yet been derandomized, and a cheater can thereby safely be identified without exposing the signing key. Once a putative correlation is generated, Doerner et al. specify a simple pairwise statistical check that the parties use to verify the correlation's correctness. We redesign this checking mechanism so that each party obtains a *public commitment* to the

[19] In other words, we *preprocess* the VOLE invocations.

private input and output shares of every other party as a byproduct. Identifying cheaters thereafter is a simple matter of finding a party who cannot prove that its commitments satisfy the correct relations.

A Simple First Attempt. An *ECDSA tuple* is a type of correlated randomness tailored to ECDSA, formalized by Abram et al. [2] after being used implicitly by Lindell and Nof [77], and also central to the protocol of Doerner et al. [48]. An ECDSA tuple comprises (ϕ, r, u, v) such that $u = \phi \cdot r$ and $v = \phi \cdot \mathsf{sk}$. Generation of a secret-shared ECDSA tuple can be accomplished via one set of symmetric pairwise VOLE instances; each party plays the role of sender once with every other party. Doerner et al. showed that this creates BeDOZa-style MACs [14] as a useful byproduct: each party's individual share of ϕ functions as its own MAC key. A shared ECDSA tuple can be verified inexpensively by checking these statistical pairwise MACs. Given shares of an ECDSA tuple, parties can sign a message non-interactively by revealing u and $w = \mathsf{SHA2}(m) \cdot \phi + r^{\mathsf{x}} \cdot v$, and then assembling $s = w/u$.

An abort can occur during the generation of the ECDSA tuple, or during assembly of the signature. These cases require qualitatively different approaches to the tracing of cheaters, as the former could potentially allow the opening of certain ephemeral secret values, whereas it is unsafe to reveal *any* secrets after signature assembly is attempted.

We first consider how to sample an ECDSA tuple with identifiability. As written, we cannot settle for *input revealing* IA, since the tuple depends upon the long-term key sk. However, given a completely random ECDSA tuple $(\phi, \tilde{r}, \tilde{r} \cdot \phi, \tilde{\mathsf{sk}} \cdot \phi)$, the parties can easily derandomize the correlation to $(\phi, r, r \cdot \phi, \mathsf{sk} \cdot \phi)$ for some chosen sk and r by publishing $\mathsf{sk} - \tilde{\mathsf{sk}}$ and $r - \tilde{r}$. This allows us to use component functionalities that have IRIA in our protocol, so long as they can never abort after derandomization has occurred.

This suggests a simple protocol template: sample a random ECDSA tuple via CVOLE and verify it just as Doerner et al. do, after which the tuple can be derandomized. If the MAC check fails, open the CVOLE to find the cheater. The latter operation is safe because the tuple is not (yet) related to any long-term secret values. This simple idea contains a subtle simulation challenge: the simulator cannot be oblivious to $\tilde{\mathsf{sk}}$ and $\tilde{r}$ (and therefore sk and r) where $\tilde{\mathsf{sk}} \cdot G$ and $\tilde{r} \cdot G$ are public, while retaining the ability to reveal $\tilde{\mathsf{sk}}$ and $\tilde{r}$ on demand if it is falsely accused of cheating by a corrupt party. This issue is reminiscent of a notoriously difficult problem in simulating discrete-logarithm-based cryptosystems against an adaptive adversary: a simulator must be oblivious to x for some public $x \cdot G$ owned by an honest party, and also prepared to reveal x in order to explain the honest party's view if it is corrupted. Here the selective opening principle[20] is key: we use SCVOLE to open *only* ϕ and check the consistency of the rest of the view, while keeping $\tilde{\mathsf{sk}}$ and $\tilde{r}$ hidden. Looking ahead, a party is given an opportunity to complain if consistency checks fails, and such a com-

[20] In Section 5, this principle allowed us to verify MACs without revealing inputs, and thereby achieve IA directly.

plaint is resolved by opening that party's share of ϕ: either an inconsistency can be identified in the behaviour of a counterparty, or the accusing party is guilty.

The Freedom of ϕ. The above strategy is enabled by the fact that ϕ is completely unspecified until the signature is assembled. This freedom is also necessary for the security analysis of any protocol that employs ECDSA tuples, since ϕ must function as a uniformly random mask for r. However, if the signature share assembly fails and a cheater must be traced, the freedom of ϕ becomes a liability. ϕ is well-defined at this stage, but no individual party is explicitly bound to its share of ϕ, making honest behaviour difficult to prove. On the other hand, ϕ cannot be revealed once signature assembly begins because a corrupt party with knowledge of w and u can compute $r = u/\phi$, and then use ϕ, r, and w to compute sk. We therefore tweak the protocol so that each party creates a Pedersen commitment [79] to its share of ϕ and proves via simple pairwise statistical checks (à la [48,14]) that the committed value corresponds to the VOLE inputs used to sample the ECDSA tuple. As a convenient byproduct, these checks yield Pedersen commitments to the VOLE outputs, and thus the protocol creates a public commitment to *every* private value held by every party. If signature assembly fails, each party can vindicate itself by proving a simple linear relation between its claimed signature share and these committed secrets in zero-knowledge. The freedom of ϕ is undisturbed, since Pedersen commitments are perfectly hiding, and simple Schnorr-like sigma protocols for discrete logarithm relations can be used for vindication. Because the commitments use the same curve as the signature, this mechanism is efficient and requires only the discrete logarithm assumption on that curve, which is a pre-requisite for ECDSA.

Putting it Together. We give an overview of each of the four phases of signing below. Let $\mathbf{P}$ be a vector containing the indices of the signing parties, and let $\mathbf{P}^{-i} = \mathbf{P} \setminus \{i\}$. We assume each party $\mathcal{P}_i$ holds an additive secret key share sk_i, and that a random base $\hat{G} \in \mathbb{G}$ has been sampled during key generation in such a way that its discrete logarithm relative to G is unknown to any party. We describe the cheater-identification mechanism afterward.

Phase 1: Each party $\mathcal{P}_i$ samples $r_i, \phi_i, \hat{\phi}_i$ from $\mathbb{Z}_q$, commits to $R_i := r_i \cdot G$, and broadcasts a Pedersen commitment $C_i^{\phi} := \phi_i \cdot G + \hat{\phi}_i \cdot \hat{G}$. Simultaneously, $\mathcal{P}_i$ initiates two instances of $[\![\mathcal{F}_{\mathsf{SCVOLE\text{-}SIRIA}}(\mathbb{Z}_q, 2)]\!]_{\mathsf{PV}}$ with each $\mathcal{P}_j$, in which it plays Bob's role and uses randomly-sampled values $\boldsymbol{\chi}^1_{j,i}$ and $\boldsymbol{\chi}^2_{j,i}$ as its inputs.

Phase 2: With each $\mathcal{P}_j$ for $j \in \mathbf{P}^{-i}$ as its counterparty, $\mathcal{P}_i$ inputs $(\phi_i, \hat{\phi}_i)$ to *both* instances of $[\![\mathcal{F}_{\mathsf{SCVOLE\text{-}SIRIA}}(\mathbb{Z}_q, 2)]\!]_{\mathsf{PV}}$ in which it plays the role of Alice, and receives $(\mathbf{t}^{\phi,1}_{i,j}, \hat{\mathbf{t}}^{\phi,1}_{i,j})$ and $(\mathbf{t}^{\phi,2}_{i,j}, \hat{\mathbf{t}}^{\phi,2}_{i,j})$ as outputs, while $\mathcal{P}_j$ receives $(\mathbf{t}^{\chi,1}_{j,i}, \hat{\mathbf{t}}^{\chi,1}_{j,i})$ and $(\mathbf{t}^{\chi,2}_{j,i}, \hat{\mathbf{t}}^{\chi,2}_{j,i})$. These values satisfy the relations

$$\mathbf{t}^{\phi,1}_{i,j} + \mathbf{t}^{\chi,1}_{j,i} = \phi_i \cdot \boldsymbol{\chi}^1_{j,i} \qquad \hat{\mathbf{t}}^{\phi,1}_{i,j} + \hat{\mathbf{t}}^{\chi,1}_{j,i} = \hat{\phi}_i \cdot \boldsymbol{\chi}^1_{j,i}$$
$$\mathbf{t}^{\phi,2}_{i,j} + \mathbf{t}^{\chi,2}_{j,i} = \phi_i \cdot \boldsymbol{\chi}^2_{j,i} \qquad \hat{\mathbf{t}}^{\phi,2}_{i,j} + \hat{\mathbf{t}}^{\chi,2}_{j,i} = \hat{\phi}_i \cdot \boldsymbol{\chi}^2_{j,i}$$

unless $\mathcal{P}_i$ used inconsistent values of $(\phi_i, \hat{\phi}_i)$ in the two instances of $[\![\mathcal{F}_{\mathsf{SCVOLE\text{-}SIRIA}}(\mathbb{Z}_q, 2)]\!]_{\mathsf{PV}}$. $\mathcal{P}_i$ proves to $\mathcal{P}_j$ that both sets of inputs correspond to the opening of C_i^{ϕ} by sending the following *check messages*:

$$\mathbf{\Gamma}_{i,j}^{\phi,1} \coloneqq \mathbf{t}_{i,j}^{\phi,1} \cdot G + \hat{\mathbf{t}}_{i,j}^{\phi,1} \cdot \hat{G} \qquad \text{and} \qquad \mathbf{\Gamma}_{i,j}^{\phi,2} \coloneqq \mathbf{t}_{i,j}^{\phi,2} \cdot G + \hat{\mathbf{t}}_{i,j}^{\phi,2} \cdot \hat{G}$$

The values $\boldsymbol{\chi}_{j,i}^1$ and $\boldsymbol{\chi}_{j,i}^2$ essentially serve as $\mathcal{P}_j$'s private MAC keys, which it can use to statistically validate $\mathcal{P}_i$'s check messages:

$$\begin{aligned} \boldsymbol{\chi}_{j,i}^1 \cdot C_i^{\phi} - \mathbf{\Gamma}_{i,j}^{\phi,1} &= \mathbf{t}_{j,i}^{\chi,1} \cdot G + \hat{\mathbf{t}}_{j,i}^{\chi,1} \cdot \hat{G} \\ \boldsymbol{\chi}_{j,i}^2 \cdot C_i^{\phi} - \mathbf{\Gamma}_{i,j}^{\phi,2} &= \mathbf{t}_{j,i}^{\chi,2} \cdot G + \hat{\mathbf{t}}_{j,i}^{\chi,2} \cdot \hat{G} \end{aligned}$$

As a convenient byproduct of this checking mechanism, $\mathbf{\Gamma}_{i,j}^{\phi,1}$ and $\mathbf{\Gamma}_{i,j}^{\phi,2}$ are Pedersen commitments to the values $\mathbf{t}_{i,j}^{\phi,1}$ and $\mathbf{t}_{i,j}^{\phi,2}$ respectively. Note that the checks we have just described are performed in both directions simultaneously.

Phase 3: If the previous checks pass, $\mathcal{P}_i$ decommits R_i and derandomizes $\boldsymbol{\chi}_{i,j}^1$ to ϕ_i and $\boldsymbol{\chi}_{i,j}^2$ to sk_i by broadcasting $\boldsymbol{\psi}_{i,j}^1 \coloneqq r_i - \boldsymbol{\chi}_{i,j}^1$ and $\boldsymbol{\psi}_{i,j}^2 \coloneqq \mathsf{sk}_i - \boldsymbol{\chi}_{i,j}^2$. Now $\mathcal{P}_i$ must prove that its derandomized SCVOLE inputs are indeed the discrete logarithms of R_i and pk_i. $\mathcal{P}_i$ broadcasts

$$\mathbf{\Gamma}_{i,j}^{r} \coloneqq \mathbf{t}_{i,j}^{\chi,1} \cdot G \qquad \text{and} \qquad \mathbf{\Gamma}_{i,j}^{\mathsf{sk}} \coloneqq \mathbf{t}_{i,j}^{\chi,2} \cdot G$$

and now ϕ_j serves as $\mathcal{P}_j$'s MAC key to check $\mathcal{P}_i$'s claim. Verifying the following equations statistically validates that $\boldsymbol{\psi}_{i,j}^1$ and $\boldsymbol{\psi}_{i,j}^2$ derandomize $\boldsymbol{\chi}_{i,j}^1$ and $\boldsymbol{\chi}_{i,j}^2$ to the discrete logarithms of R_i and pk_i, respectively:

$$\begin{aligned} \phi_j \cdot \bigl(R_i - \boldsymbol{\psi}_{i,j}^1 \cdot G\bigr) - \mathbf{\Gamma}_{i,j}^{r} &= \mathbf{t}_{j,i}^{\phi,1} \cdot G \\ \phi_j \cdot \bigl(\mathsf{pk}_i - \boldsymbol{\psi}_{i,j}^2 \cdot G\bigr) - \mathbf{\Gamma}_{i,j}^{r} &= \mathbf{t}_{j,i}^{\phi,2} \cdot G \end{aligned}$$

At the end of this phase, the common nonce $R \coloneqq \sum_{i \in \mathbf{P}} R_i$ is established, and every party's secrets are committed publicly: besides pk_i and R_i being perfectly binding commitments to sk_i and r_i, the byproducts of our checking mechanism C_i^{ϕ}, $\mathbf{\Gamma}_{i,j}^{\phi,1}$, $\mathbf{\Gamma}_{i,j}^{\phi,2}$, $\mathbf{\Gamma}_{i,j}^{r}$, and $\mathbf{\Gamma}_{i,j}^{\mathsf{sk}}$ are public commitments to $\mathcal{P}_i$'s secrets ϕ_i, $\boldsymbol{\chi}_{i,j}^1$, $\boldsymbol{\chi}_{i,j}^2$, $\mathbf{t}_{i,j}^{\chi,1}$, and $\mathbf{t}_{i,j}^{\chi,2}$ respectively.

Phase 4: If the previous checks pass, each $\mathcal{P}_i$ computes

$$\begin{aligned} u_i &\coloneqq \phi_i \cdot \Bigl(r_i + \sum_{j \in \mathbf{P}^{-i}} \boldsymbol{\psi}_{j,i}^1\Bigr) + \sum_{j \in \mathbf{P}^{-i}} (\mathbf{t}_{i,j}^{\phi,1} + \mathbf{t}_{i,j}^{\chi,1}) \\ v_i &\coloneqq \phi_i \cdot \Bigl(\mathsf{sk}_i + \sum_{j \in \mathbf{P}^{-i}} \boldsymbol{\psi}_{j,i}^2\Bigr) + \sum_{j \in \mathbf{P}^{-i}} (\mathbf{t}_{i,j}^{\phi,2} + \mathbf{t}_{i,j}^{\chi,2}) \\ w_i &\coloneqq \mathsf{SHA2}(m) \cdot \phi_i + r^{\mathsf{x}} \cdot v_i \end{aligned}$$

and broadcasts u_i and w_i so that the signature $(R, s \coloneqq (\sum_i w_i)/(\sum_i u_i))$ can be assembled publicly.

Cheater Identification. The above protocol can fail in two ways: either a malformed protocol message triggers a complaint in any of the phases, or Phase 4 results in an invalid signature. These are handled through different mechanisms.

1. Any complaint between a pair of parties concerning a VOLE instance or a failure of the checks can be resolved by opening Alice's side of $[\![\mathcal{F}_{\mathsf{SCVOLE\text{-}SIRIA}}(\mathbb{Z}_q, 2)]\!]_{\mathsf{PV}}$. If $\mathcal{P}_i$ plays Alice, then this reveals ϕ_i, $\hat{\phi}_i$, $\mathbf{t}^{\phi,1}_{i,j}$, $\hat{\mathbf{t}}^{\phi,1}_{i,j}$, $\mathbf{t}^{\phi,2}_{i,j}$, $\hat{\mathbf{t}}^{\phi,2}_{i,j}$, after which all parties can verify their relationships to C^{ϕ}_{i}, $\mathbf{\Gamma}^{\phi,1}_{i,j}$, $\mathbf{\Gamma}^{\phi,2}_{i,j}$, $\mathbf{\Gamma}^{r}_{i,j}$, and $\mathbf{\Gamma}^{\mathsf{sk}}_{i,j}$. If no inconsistency is detected, the complainer is deemed to be corrupt.
2. If there is no explicit complaint, but signature assembly fails in the final phase, $[\![\mathcal{F}_{\mathsf{SCVOLE\text{-}SIRIA}}(\mathbb{Z}_q, 2)]\!]_{\mathsf{PV}}$ cannot be opened because revealing ϕ would compromise sk. However, the lack of complaints indicates that all secret values are available in publicly-committed form. If each $\mathcal{P}_i$ additionally provides Pedersen commitments $C^{\phi r}_{i}$ and $C^{\phi \mathsf{sk}}_{i}$ to the products $\phi_i \cdot r_i$ and $\phi_i \cdot \mathsf{sk}_i$, then $\mathcal{P}_i$'s computation of u_i and w_i can be publicly replicated in committed form as a linear combination of its committed secrets as follows:

   $$\begin{aligned} \text{An honest } \mathcal{P}_i \text{ computed } \; u_i &= \phi_i r_i &&+ \sum_{j \in \mathbf{P}^{\text{-}i}} \boldsymbol{\psi}^{1}_{j,i} \cdot \phi_i &&+ \sum_{j \in \mathbf{P}^{\text{-}i}} (\mathbf{t}^{\phi,1}_{i,j} + \mathbf{t}^{\chi,1}_{i,j}) \\ \text{Any verifier can derive } \; U_i &= C^{\phi r}_{i} &&+ \sum_{j \in \mathbf{P}^{\text{-}i}} \boldsymbol{\psi}^{1}_{j,i} \cdot C^{\phi}_{i} &&+ \sum_{j \in \mathbf{P}^{\text{-}i}} (\mathbf{\Gamma}^{\phi,1}_{i,j} + \mathbf{\Gamma}^{r}_{i,j}) \end{aligned}$$

   Simplifying the above term yields $U_i = u_i \cdot G + \rho_i \cdot \hat{G}$ where ρ_i is the sum of the Pedersen commitment randomness for C^{ϕ}_{i}, $C^{\phi r}_{i}$, and $\mathbf{\Gamma}^{\phi,1}_{i,j}$. Since U_i is essentially a Pedersen commitment to u_i, we have $U_i - u_i \cdot G = \rho_i \cdot \hat{G}$, which we can interpret as a Pedersen commitment to zero. Similarly, if $\mathcal{P}_i$ supplies $C^{\phi \mathsf{sk}}_{i}$ as a commitment to $\phi_i \cdot \mathsf{sk}_i$, it facilitates the public derivation of W_i as a commitment to w_i. Therefore, in order for $\mathcal{P}_i$ to prove that the u_i and w_i it broadcasted in Phase 4 were honestly derived relative to its publicly committed secrets, it suffices for $\mathcal{P}_i$ to prove in zero-knowledge that it knows the discrete logarithms of $U_i - u_i \cdot G$ and $W_i - w_i \cdot G$ relative to $\hat{G}$, and that $C^{\phi r}_{i}$ commits to the product of the opening of C^{ϕ}_{i} and the discrete logarithm of R_i, and that the same relation holds for $C^{\phi \mathsf{sk}}_{i}$, C^{ϕ}_{i}, and pk_i.
   There are simple Schnorr-like Sigma protocols for this task, which can be compiled into simulation-extractable NIZKs via the Fiat-Shamir transform. Straight-line extraction is *not* required of the NIZKs in this protocol because every component of the witness is already available to the simulator as an input to $[\![\mathcal{F}_{\mathsf{SCVOLE\text{-}SIRIA}}(\mathbb{Z}_q, 2)]\!]_{\mathsf{PV}}$. The extractor of the NIZK is only invoked in the reduction, in order to construct an algorithm that outputs the discrete logarithm of $\hat{G}$ relative to G if the adversary is able to cheat undetected but still produce a proof of honest behaviour.

We refer the reader to the full version for a formal specification of the protocol.

Signing efficiency. Much like VOLE is the dominant cost in the unmodified protocol of Doerner et al. [48], SCVOLE dominates our signing protocol for standard parameters (i.e., a 256-bit elliptic curve and 80-bits of statistical security). Whereas their work uses a single invocation of $\mathcal{F}_{\mathsf{RVOLE}}(q, 2)$ in each direction between each pair of parties, our protocol uses two invocations of $[\![\mathcal{F}_{\mathsf{SCVOLE\text{-}SIRIA}}(\mathbb{Z}_q, 2)]\!]_{\mathsf{PV}}$. As shown in the full version [31], the bandwidth cost of instantiating $[\![\mathcal{F}_{\mathsf{SCVOLE\text{-}SIRIA}}(\mathbb{Z}_q, 2)]\!]_{\mathsf{PV}}$ is roughly 150% of instantiating $\mathcal{F}_{\mathsf{RVOLE}}(q, 2)$; thus the overall bandwidth of our ECDSA-IA protocol is roughly triple that of Doerner et al.'s protocol. In terms of round count, Phases 1 and 2 are dominated by SCVOLE, which requires 5 rounds as per the full version, and Phases 3 and 4 require one round each, bringing the total to 7 rounds. This is better than many *deployed* threshold ECDSA signing protocols [55,47,77].

Acknowledgements

Ran Cohen's research was supported in part by NSF grant no. 2055568, by ISF grant 1834/23, and by the Algorand Centres of Excellence programme managed by Algorand Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Algorand Foundation. Jack Doerner's research was supported in part by the ERC under projects NTSC (742754) and HSS (852952), by ISF grant 2774/2, by AFOSR award FA9550-21-1-0046, by the Azrieli Foundation, and by the Brown University Data Science Initiative. Yashvanth Kondi's research was supported in part by the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM) and by the ERC under project SPEC (803096). abhi shelat's research was supported in part by the NSF under grants 1646671 and 1816028.

References

1. Damiano Abram, Jack Doerner, Yuval Ishai, and Varun Narayanan. Constant-round simulation-secure coin tossing extension with guaranteed output. In *Advances in Cryptology – EUROCRYPT 2024, part V*, pages 122–154, 2024.
2. Damiano Abram, Ariel Nof, Claudio Orlandi, Peter Scholl, and Omer Shlomovits. Low-bandwidth threshold ECDSA via pseudorandom correlation generators. In *Proceedings of the 43rd IEEE Symposium on Security and Privacy, (S&P)*, 2022.
3. Bar Alon, Hao Chung, Kai-Min Chung, Mi-Ying Huang, Yi Lee, and Yu-Ching Shen. Round efficient secure multiparty quantum computation with identifiable abort. In *Advances in Cryptology – CRYPTO 2021, part I*, pages 436–466, 2021.
4. Bar Alon and Eran Omri. Almost-optimally fair multiparty coin-tossing with nearly three-quarters malicious. In *Proceedings of the 14th Theory of Cryptography Conference, TCC 2016-B, part I*, pages 307–335, 2016.
5. Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *Journal of Cryptology*, 23(2):281–343, 2010.
6. Carsten Baum, Nikolas Melissaris, Rahul Rachuri, and Peter Scholl. Cheater identification on a budget: Mpc with identifiable abort from pairwise macs. Cryptology ePrint Archive, Paper 2023/1548, 2023. https://eprint.iacr.org/2023/1548.

7. Carsten Baum, Emmanuela Orsini, and Peter Scholl. Efficient secure multiparty computation with identifiable abort. In *Proceedings of the 14th Theory of Cryptography Conference, TCC 2016-B, part I*, pages 461–490, 2016.
8. Carsten Baum, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. Efficient constant-round MPC with identifiable abort and public verifiability. In *Advances in Cryptology – CRYPTO 2020, part II*, pages 562–592, 2020.
9. Donald Beaver. Precomputing oblivious transfer. In *Advances in Cryptology – CRYPTO 1995*, pages 97–109, 1995.
10. Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 503–513, 1990.
11. Amos Beimel, Yehuda Lindell, Eran Omri, and Ilan Orlov. $1/p$-secure multiparty computation without honest majority and the best of both worlds. In *Advances in Cryptology – CRYPTO 2011*, pages 277–296, 2011.
12. Amos Beimel, Eran Omri, and Ilan Orlov. Protocols for multiparty coin toss with a dishonest majority. *Journal of Cryptology*, 28(3):551–600, 2015.
13. Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *Proceedings of the 36th IEEE Symposium on Security and Privacy, (S&P)*, pages 287–304, 2015.
14. Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Advances in Cryptology – EUROCRYPT 2011*, pages 169–188, 2011.
15. Dan Boneh and Matthew K. Franklin. Efficient generation of shared RSA keys. *Journal of the ACM*, 48(4):702–722, 2001.
16. Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-snark parameters in the random beacon model. *IACR Cryptol. ePrint Arch.*, 2017:1050, 2017.
17. Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Secure multiparty computation with sublinear preprocessing. In *Advances in Cryptology – EUROCRYPT 2022, part I*, pages 427–457, 2022.
18. Nicholas Brandt. Tight setup bounds for identifiable abort. http://eprint.iacr.org/2021/684, 2021.
19. Nicholas-Philip Brandt, Sven Maier, Tobias Müller, and Jörn Müller-Quade. Constructing secure multi-party computation with identifiable abort. http://eprint.iacr.org/2020/153, 2020.
20. Niv Buchbinder, Iftach Haitner, Nissan Levi, and Eliad Tsfadia. Fair coin flipping: Tighter analysis and the many-party case. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2580–2600, 2017.
21. Jakob Burkhardt, Ivan Damgård, Tore Kasper Frederiksen, Satrajit Ghosh, and Claudio Orlandi. Improved distributed RSA key generation using the miller-rabin test. *IACR Cryptol. ePrint Arch.*, page 644, 2023.
22. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2001.
23. Ran Canetti and Marc Fischlin. Universally composable commitments. In *Advances in Cryptology – CRYPTO 2001*, pages 19–40, 2001.
24. Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In *Proceedings of the 27th ACM Conference on Computer and Communications Security, (CCS)*, pages 1769–1787. ACM, 2020.

25. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 494–503, 2002.
26. Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold EC-DSA revisited: Online/offline extensions, identifiable aborts proactive and adaptive security. *Theor. Comput. Sci.*, 939:78–104, 2023.
27. Megan Chen, Ran Cohen, Jack Doerner, Yashvanth Kondi, Eysa Lee, Schuyler Rosefield, and abhi shelat. Multiparty generation of an RSA modulus. In *Advances in Cryptology – CRYPTO 2020, part III*, pages 64–93, 2020.
28. Megan Chen, Carmit Hazay, Yuval Ishai, Yuriy Kashnikov, Daniele Micciancio, Tarik Riviere, abhi shelat, Muthu Venkitasubramaniam, and Ruihan Wang. Diogenes: Lightweight scalable RSA modulus generation with a dishonest majority. In *Proceedings of the 42nd IEEE Symposium on Security and Privacy, (S&P)*, pages 590–607, 2021.
29. Michele Ciampi, Divya Ravi, Luisa Siniscalchi, and Hendrik Waldner. Round-optimal multi-party computation with identifiable abort. In *Advances in Cryptology – EUROCRYPT 2022, part I*, pages 335–364, 2022.
30. Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 364–369, 1986.
31. Ran Cohen, Jack Doerner, Yashvanth Kondi, and abhi shelat. Secure multiparty computation with identifiable abort via vindicating release. Cryptology ePrint Archive, Paper 2023/1136, 2023. https://eprint.iacr.org/2023/1136.
32. Ran Cohen, Jack Doerner, Yashvanth Kondi, and abhi shelat. Guaranteed output in $O(\sqrt{n})$ rounds for round-robin sampling protocols. In *Advances in Cryptology – EUROCRYPT 2022, part I*, pages 241–271, 2022.
33. Ran Cohen, Juan A. Garay, and Vassilis Zikas. Broadcast-optimal two-round MPC. In *Advances in Cryptology – EUROCRYPT 2020, part II*, pages 828–858, 2020.
34. Ran Cohen, Iftach Haitner, Eran Omri, and Lior Rotem. From fairness to full security in multiparty computation. *Journal of Cryptology*, 35(1):4, 2022.
35. Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. *Journal of Cryptology*, 30(4):1157–1186, 2017.
36. Claude Crépeau, Jeroen van de Graaf, and Alain Tapp. Committed oblivious transfer and private multi-party computation. In *Advances in Cryptology – CRYPTO 1995*, pages 110–123, 1995.
37. Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-interactive and non-malleable commitment. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 141–150, 1998.
38. Robert K. Cunningham, Benjamin Fuller, and Sophia Yakoubov. Catching MPC cheaters: Identification and openability. In *Proceedings of the 10th International Conference on Information Theoretic Security (ICITS)*, pages 110–134, 2017.
39. Dana Dachman-Soled. Revisiting fairness in MPC: polynomial number of parties and general adversarial structures. In *Proceedings of the 18th Theory of Cryptography Conference, TCC 2020, part II*, pages 595–620, 2020.
40. Ivan Damgård. On σ-protocols. https://www.cs.au.dk/~ivan/Sigma.pdf, 2010.
41. Ivan Damgård, Bernardo Magri, Divya Ravi, Luisa Siniscalchi, and Sophia Yakoubov. Broadcast-optimal two round MPC with an honest majority. In *Advances in Cryptology – CRYPTO 2021, part II*, pages 155–184, 2021.

42. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology – CRYPTO 2012*, pages 643–662, 2012.
43. Ivan Damgård, Divya Ravi, Luisa Siniscalchi, and Sophia Yakoubov. Minimizing setup in broadcast-optimal two round MPC. http://eprint.iacr.org/2022/293, 2022.
44. Cyprien Delpech de Saint Guilhem, Eleftheria Makri, Dragos Rotaru, and Titouan Tanguy. The return of eratosthenes: Secure generation of RSA moduli using distributed sieving. In *Proceedings of the 28th ACM Conference on Computer and Communications Security, (CCS)*, pages 594–609, 2021.
45. Yevgeniy Dodis, Victor Shoup, and Shabsi Walfish. Efficient constructions of composable commitments and zero-knowledge proofs. In *Advances in Cryptology – CRYPTO 2008*, pages 515–535, 2008.
46. Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. Secure two-party threshold ECDSA from ECDSA assumptions. In *Proceedings of the 39th IEEE Symposium on Security and Privacy, (S&P)*, pages 980–997, 2018.
47. Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *Proceedings of the 40th IEEE Symposium on Security and Privacy, (S&P)*, 2019.
48. Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. Threshold ecdsa in three rounds. Cryptology ePrint Archive, Paper 2023/765, 2023. https://eprint.iacr.org/2023/765.
49. Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.
50. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO 1986*, pages 186–194, 1986.
51. Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *Advances in Cryptology – CRYPTO 2005*, pages 152–168, 2005.
52. Jun Furukawa, Yehuda Lindell, Ariel Nof, and Or Weinstein. High-throughput secure three-party computation for malicious adversaries and an honest majority. In *Advances in Cryptology – EUROCRYPT 2017, part II*, pages 225–255, 2017.
53. Chaya Ganesh, Yashvanth Kondi, Arpita Patra, and Pratik Sarkar. Efficient adaptively secure zero-knowledge from garbled circuits. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 499–529. Springer, 2018.
54. Juan A. Garay. Efficient and universally composable committed oblivious transfer and applications. In *Proceedings of the First Theory of Cryptography Conference, TCC 2004*, pages 297–316, 2004.
55. Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1179–1194. ACM, 2018.
56. Niv Gilboa. Two party RSA key generation. In *Advances in Cryptology – CRYPTO 1999*, pages 116–129, 1999.
57. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the*

19th Annual ACM Symposium on Theory of Computing (STOC), pages 218–229, 1987.

58. S. Dov Gordon and Jonathan Katz. Complete fairness in multi-party computation without an honest majority. In *Proceedings of the 6th Theory of Cryptography Conference, TCC 2009*, pages 19–35, 2009.
59. S. Dov Gordon and Jonathan Katz. Partial fairness in secure two-party computation. *Journal of Cryptology*, 25(1):14–40, 2012.
60. Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-snarks. In *Advances in Cryptology – CRYPTO 2018, part III*, pages 698–728, 2018.
61. Iftach Haitner and Eliad Tsfadia. An almost-optimally fair three-party coin-flipping protocol. *SIAM Journal on Computing*, 46(2):479–542, 2017.
62. Carmit Hazay, Gert Læssøe Mikkelsen, Tal Rabin, and Tomas Toft. Efficient RSA key generation and threshold Paillier in the two-party setting. In *Proceedings of the Cryptographers' Track at the RSA Conference (CT-RSA)*, pages 313–331, 2012.
63. Carmit Hazay, Peter Scholl, and Eduardo Soria-Vazquez. Low cost constant round MPC combining BMR and oblivious transfer. In *Advances in Cryptology – ASIACRYPT 2017, part I*, pages 598–628, 2017.
64. Martin Hirt, Ueli Maurer, and Vassilis Zikas. MPC vs. SFE : Unconditional and computational security. In *Advances in Cryptology – ASIACRYPT 2008*, pages 1–18, 2008.
65. Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 12–24, 1989.
66. Yuval Ishai, Jonathan Katz, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On achieving the "best of both worlds" in secure multiparty computation. *SIAM Journal on Computing*, 40(1):122–141, 2011.
67. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 21–30, 2007.
68. Yuval Ishai, Rafail Ostrovsky, and Hakan Seyalioglu. Identifying cheaters without an honest majority. In *Proceedings of the 9th Theory of Cryptography Conference, TCC 2012*, pages 21–38, 2012.
69. Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In *Advances in Cryptology – CRYPTO 2014, part II*, pages 369–386, 2014.
70. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *Advances in Cryptology – CRYPTO 2008*, pages 572–591, 2008.
71. Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *Proceedings of the 20th ACM Conference on Computer and Communications Security, (CCS)*, pages 955–966. ACM, 2013.
72. Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In *Proceedings of the 10th Theory of Cryptography Conference, TCC 2013*, pages 477–498, 2013.
73. Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 23th ACM Conference on Computer and Communications Security, (CCS)*, pages 830–842, 2016.

74. Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 20–31, 1988.
75. Markulf Kohlweiss, Mary Maller, Janno Siim, and Mikhail Volkhov. Snarky ceremonies. In *Advances in Cryptology – ASIACRYPT 2021, part III*, pages 98–127, 2021.
76. Yashvanth Kondi and abhi shelat. Improved straight-line extraction in the random oracle model with applications to signature aggregation. In *Advances in Cryptology – ASIACRYPT 2022, part II*, pages 279–309, 2022.
77. Yehuda Lindell and Ariel Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In *Proceedings of the 25th ACM Conference on Computer and Communications Security, (CCS)*, pages 1837–1854, 2018.
78. Rafael Pass. On deniability in the common reference string and random oracle model. In *Advances in Cryptology – CRYPTO 2003*, pages 316–337, 2003.
79. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO 1991*, pages 129–140, 1991.
80. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology – CRYPTO 2008*, pages 554–571, 2008.
81. Willy Quach. Uc-secure OT from lwe, revisited. In *Security and Cryptography for Networks - 12th International Conference, SCN 2020, Amalfi, Italy, September 14-16, 2020, Proceedings*, pages 192–211, 2020.
82. Lawrence Roy. Softspokenot: Quieter OT extension from small-field silent VOLE in the minicrypt model. In *Advances in Cryptology – CRYPTO 2022, part I*, pages 657–687, 2022.
83. Mark Simkin, Luisa Siniscalchi, and Sophia Yakoubov. On sufficient oracles for secure computation with identifiable abort. In *Proceedings of the 13th Conference on Security and Cryptography for Networks (SCN)*, pages 494–515, 2022.
84. Nigel P. Smart and Titouan Tanguy. Taas: Commodity MPC via triples-as-a-service. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop, CCSW@CCS 2019, London, UK, November 11, 2019*, pages 105–116, 2019.
85. Gabriele Spini and Serge Fehr. Cheater detection in SPDZ multiparty computation. In *Proceedings of the 9th International Conference on Information Theoretic Security (ICITS)*, pages 151–176, 2016.
86. Xiao Wang, Samuel Ranellucci, and John Katz. Global-scale secure multiparty computation. In *Proceedings of the 24th ACM Conference on Computer and Communications Security, (CCS)*, pages 39–56, 2017.
87. Kang Yang, Xiao Wang, and Jiang Zhang. More efficient MPC from improved triple generation and authenticated garbling. In *Proceedings of the 27th ACM Conference on Computer and Communications Security, (CCS)*, pages 1627–1646, 2020.
88. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, 1982.
89. Vassilis Zikas, Sarah Hauser, and Ueli Maurer. Realistic failures in secure multiparty computation. In *Proceedings of the 6th Theory of Cryptography Conference, TCC 2009*, pages 274–293, 2009.