Putting the OPTI in Round Optimal IA-MPC in the Plain Model

Yashvanth Kondi[1], Divya Ravi[2], Jure Sternad[3], and Sophia Yakoubov[3]

[1] Silence Laboratories, Singapore; yash@ykondi.net
[2] University of Amsterdam, Netherlands; d.ravi@uva.nl
[3] Aarhus University, Denmark; {jsternad, sophia.yakoubov}@cs.au.dk

Abstract. We advance the study of best-possible security guarantees of two-round secure multiparty computation in the plain model.
While Goel et al. (TCC '21) showed the infeasibility of Identifiable Abort in two rounds in the plain model with a standard honest majority (i.e. $t < n/2$), the effect of relaxing the setting to an honest supermajority has not been established. In this work, we close this gap.
When $t < n/3$, we show that two rounds *are* sufficient to achieve identifiable abort. Notably, ours is the first two-round general-purpose MPC construction in the plain model tolerating more than one corruption that achieves a guarantee stronger than unanimous abort.
We obtain our construction via a new compiler that lifts any protocol with unanimous abort to one that achieves identifiable abort. A core building block for our compiler is *Oblivious Public Transfer with Intermediaries* (OPTI), a new primitive that we introduce which may be of independent interest.

Table of Contents

Putting the OPTI in Round Optimal IA-MPC in the Plain Model 1
Yashvanth Kondi, Divya Ravi, Jure Sternad, and Sophia Yakoubov
1 Introduction 2
1.1 Related Work 3
1.2 Our Results 4
1.3 Technical Overview 4
2 Preliminaries 7
2.1 Public Key Encryption 7
2.2 Commitment Scheme 9
2.3 Secret Sharing 10
2.4 Multi-string Non-Interactive Zero-knowledge Proofs 11
2.5 Garbling Scheme 15
3 Oblivious Public Transfer with Intermediaries (OPTI) 16
3.1 OPTI Definitions 16
3.2 OPTI Protocol 21
3.3 Communication Complexity 28
4 Definition of MPC with Identifiable Abort 29
5 Construction of Two-Round MPC with IA for $t < \frac{n}{3}$ 31

1 Introduction

Secure Multiparty Computation (MPC, [CDvdG88,GMW87,Yao86]) allows multiple parties, each with a private input, to compute a joint function of their private inputs without revealing anything about their inputs apart from the function output. MPC has been an active area of research for over four decades now. A key goal has been making MPC protocols more *secure* and more *efficient*, while reducing the infrastructure it relies on (*setup*).

Security. The security of MPC protocols is measured in two crucial ways:

Adversary's Corruption Power: How many of the n parties can an adversary control (that is, what is the corruption threshold t)?
Security Guarantee: Which outcomes can an adversary induce?

The most-studied security guarantees are as follows, roughly from weakest to strongest:

Selective Abort (SA): The adversary may deprive some of the honest parties of output.

Unanimous Abort (UA): The adversary may deprive the honest parties of output, but if it does, the honest parties all agree that the computation aborted. (Any protocol that achieves UA also achieves SA.)
Identifiable Abort (IA) [AL10]: The adversary may deprive the honest parties of output, but if it does, the honest parties all agree that the computation aborted, and they are able to unanimously identify at least one corrupt party.
Fairness (FAIR): The adversary may deprive the honest parties of output, but if it does, the adversary does not learn the output either. (Any protocol that achieves fairness also achieves UA.)
Guaranteed Output Delivery (GOD): The adversary may not deprive the honest parties of output. (Any protocol that achieves GOD also achieves all other guarantees.)

Efficiency. There are many axes along which the efficiency of MPC protocols is measured; among the most-studied are communication complexity and round complexity. General-purpose MPC protocols are known to require at least two rounds of communication [HLP11]. In this paper, we focus on protocols that run in exactly two rounds.

Setup. Some MPC protocols rely on a common reference string (CRS); others require a public key infrastructure (PKI), where all participants know one another's public keys ahead of time; yet others use an even more involved form of setup, where participants hold *correlated randomness*. In this paper, we focus on protocols that require *no* setup. This is known as the *plain model*.

In this paper, we advance the study of what security is achievable in two rounds in the plain model, for various values of the corruption threshold t.

1.1 Related Work

As noted earlier, two rounds are necessary in all settings [HLP11], since one-round protocols remain vulnerable to residual attacks. The sufficiency of two rounds has been demonstrated in the presence of setup assumptions such as a CRS and/or PKI through a sequence of works [GGHR14,MW16,GS18,BL18,GLS15]. Round optimality with perfect security was investigated in [AKP20,ABT21], while the optimal round complexity of statistically secure MPC was established in [AKP23], and for NC^1 functionalities in [ACGJ19].

Below, we outline the works most relevant to our setting, namely, plain model and computational security.

When an adversary might control a majority of participants ($t \geq \frac{n}{2}$), four rounds are necessary for any security guarantee [GMPP16] without any setup. The sufficiency of four rounds for security with abort was established in a series of works [KO04,BHP17,ACJ17,CCG+20]. Four rounds are sufficient for security with IA as well [CRSW22], which is the best notion one can hope for in the dishonest majority setting: as fairness and GOD are known to be impossible without an honest majority irrespective of the number of rounds [Cle86].

The focus of our work is the honest-majority setting. When a majority of participants is guaranteed to be honest ($t < \frac{n}{2}$), UA (and thus SA) remain achievable in two rounds [ACGJ18,ABT19]. In contrast, IA is not achievable in two rounds [GJPR21], and neither are fairness or GOD [PR18,GLS15].

In the case of what is known as an honest "supermajority" ($t < \frac{n}{3}$), fairness and GOD are still unachievable in two rounds as long as $t > 1$ [GIKR02], SA [IKP10] and UA is still achievable (the construction of [ACGJ18] naturally still works if fewer parties are corrupt).

There is a crucial gap in the landscape, which we resolve in this paper.

1.2 Our Results

Prior to this work, it was unknown whether IA is achievable in two rounds in the plain model if the adversary controls $t < \frac{n}{3}$ participants. We show that it *is* (Theorem 2). Our construction is the first two-round general-purpose MPC construction in the plain model tolerating more than one corruption to achieve any guarantee stronger than unanimous abort.

Table 1 summarizes our results (positive results highlighted in green, negative in orange), as well as the rest of the landscape described above.

Threshold	Selective Abort (SA)	Unanimous Abort (UA)	Identifiable Abort (IA)	Fairness (FAIR)	Guaranteed Output Delivery (GOD)
$\frac{n}{3} \leq t < \frac{n}{2}$	✓ ⟵	✓[ACGJ18] [ABT19]	✗[GJPR21]	✗[PR18,GLS15]	⟶ ✗
$t < \frac{n}{3}$	✓ ⟵	✓	✓ (Thm. 2)	✗ for $t > 1$ ⟶ ✗ for $t > 1$ [GIKR02]	

Table 1: A map of the feasibility and impossibility of two-round MPC with different guarantees, organized by thresholds.

1.3 Technical Overview

In Section 5, we construct a protocol that computes any function f with identifiable abort in two rounds in the plain model if $t < \frac{n}{3}$.

Oblivious Public Transfer with Intermediaries (OPTI). As a building block, we define a new primitive which we call Oblivious Public Transfer with Intermediaries (OPTI). OPTI is a twist on Oblivious Transfer (OT), where a sender

supplies two inputs x_0 and x_1, and a receiver supplies a choice bit b and receives x_b as output. The security properties of OT are that the sender should not learn b, and the receiver should not learn x_{1-b}.

Like OT, OPTI has a sender and a receiver where the sender supplies x_0 and x_1, and the receiver supplies b. Unlike OT, OPTI is run among n parties, where in addition to the sender and receiver, there are $n-2$ parties who provide no input (we call these *intermediaries*). Also unlike OT, OPTI has *public output*, meaning that everyone learns x_b. This means that if $x_0 \neq x_1$, the sender naturally learns b; however, OPTI requires that if neither the sender nor receiver are corrupt, the adversary should not learn b (privacy), and if the sender is not corrupt, the adversary should not learn x_{1-b} (sender-privacy).

In addition to privacy and sender-privacy, OPTI demands *identifiability*, meaning that if honest parties don't get the right output they should be able to blame a corrupt sender or receiver.

Using OPTI To Build Secure Computation with Identifiable Abort. We use a two-round OPTI protocol that achieves security in the presence of fewer than $\frac{n}{3}$ corruptions to build a two-round secure computation protocol Π_{IA} which achieves security with identifiable abort in the same setting.

We start with a two-round secure computation protocol Π_{UA} that achieves unanimous abort in this setting [ACGJ18]. In order to turn Π_{UA} identifiable, we add zero-knowledge proofs of correct behavior. Because we are in the plain model, instead of using non-interactive zero-knowledge proofs (NIZKs), which require a common reference string (CRS), we use *multi-string* NIZKs (Section 2.4), where each participant generates their own CRS. Parties create their proofs with respect to the entire set of CRSs, and the proofs are secure as long as a majority of the CRSs were generated honestly. Our participants publish their CRSs in the first round, and publish the proofs of correct behavior in the second round.

If Π_{UA} only used public channels, we would be done; however, Π_{UA} relies on private messages between participants, and it is impossible — without further modification — for a party to publicly prove that it sent another party a correct private message, in the absence of public encryption keys. Since we operate in the plain model, parties can generate and publish encryption keys in the first round (thus eliminating the need for private channels in the second round), but this doesn't help with first-round private communication.

To address this, in Π_{IA}, instead of running Π_{UA} directly, each party garbles (Section 2.5) the function that computes its second-round messages in Π_{UA} from its state and incoming first-round messages. For each pair of input wire labels corresponding to a bit of incoming first-round message, the garbler plays the role of sender in OPTI; the sender of the corresponding first-round message bit plays the role of receiver, and everyone learns the label. The garbler broadcasts the circuit, together with labels for input wires corresponding to its local state. The circuits are then publicly evaluated to obtain all parties' second-round messages, which in turn can be used to learn the output.

The multi-string NIZKs of correct behavior are modified to prove correct garbling, and those NIZKs — together with the identifiability of OPTI — are

enough to guarantee that if honest parties do not learn the correct output, they will identify a corrupt party.

Realizing OPTI. We build a two-round OPTI protocol in the plain model by having the receiver Shamir share its choice bit with degree t in the first round. The sender does not directly share its inputs. Instead, it samples and Shamir shares two random ElGamal secret keys, one for each input, together with a degree-$2t$ sharing of zero. In the second round, the parties homomorphically combine these shares using the receiver's shares and reconstruct the key corresponding to the receiver's choice. The sender also publishes the two inputs encrypted under their corresponding public keys, in a randomly permuted order, allowing everyone to decrypt the selected input without revealing the receiver's choice to the intermediaries.

To enable identification, the sender broadcasts commitments to its two inputs, commits to its individual shares of the two random keys and the zero-sharing, and privately distributes the corresponding openings. In the second round, it uses multi-string NIZKs to prove that the sharings are valid and that the two public ciphertexts encrypt the committed inputs under the corresponding shared keys. The receiver instead commits to the coefficients of its degree-t sharing polynomial B, whose constant coefficient is $B(0) = b$. It privately sends each party $\mathcal{P}_k$ a verifiable evaluation pair (β_k, ρ_k), where $\beta_k = B(k)$ and ρ_k is the corresponding evaluation of the randomizing polynomial used in the coefficient commitment. Each party checks that its pair is consistent with the public coefficient commitments. In the second round, the receiver uses a multi-string NIZK to prove that the coefficient commitment is well-formed and that $B(0) \in \{0, 1\}$.

One issue that comes up is that some parties may (claim to) not have received valid key shares, zero-shares, and corresponding openings from the sender, or valid evaluation pairs from the receiver over private channels in the first round[4]. If more than t parties claim not to have received valid messages from someone, that is enough to identify that party as corrupt. However, it could be that t parties claim that they did not receive valid messages from the sender, while a different t parties claim that they did not receive valid messages from the receiver, which does not allow either party to be identified. Those parties cannot contribute the degree-$2t$ shares needed to reconstruct the selected key. At the tight threshold $n = 3t + 1$, this leaves only $t + 1$ shares, whereas reconstructing a degree-$2t$ sharing requires $2t + 1$ shares.

To solve this, we use the fact that if the sender is corrupt, the adversary is already allowed to know both of the sender's inputs. If a party does not receive valid shares of the two random keys and the zero-sharing from the sender in the first round, it facilitates the public recovery of those shares. Towards this public recovery, in the first round, each party broadcasts a recovery public encryption key. In the second round, the sender encrypts the corresponding key shares and

[4]If parties knew one another's public encryption keys, they could provably send private messages. However, since we operate in the plain model, the parties do not have known public keys in the first round.

zero-share under that key and broadcasts the ciphertexts, together with proofs that they contain the values fixed by the first-round commitments.

Since this enables decryption only after the second round, it comes too late for $\mathcal{P}_k$ to locally compute and broadcast its share of the selected key. However, if $\mathcal{P}_k$ did not receive valid sender shares in the first round but did receive a valid evaluation pair from the receiver, it broadcasts its recovery secret key and the pair (β_k, ρ_k) in the second round, before receiving the sender's ciphertexts. Everyone can verify the receiver's evaluation pair, decrypt the sender's key shares and zero-share, and publicly compute the missing degree-$2t$ share.

Computational Assumptions. Our construction relies on standard cryptographic primitives. The OPTI protocol uses information-theoretic secret sharing, non-interactive commitments, public-key encryption, and multi-string NIZKs, while the MPC compiler additionally uses a garbling scheme. In our current instantiation, the commitment and encryption schemes are ElGamal-based and rely on the DDH assumption, the multi-string NIZK scheme can be instantiated from enhanced trapdoor permutations, and the garbling scheme can be constructed from one-way functions.

This leads to the following concrete instantiation result: assuming DDH and the existence of enhanced trapdoor permutations, there exists a two-round MPC protocol with identifiable abort in the plain model for $t < n/3$.

2 Preliminaries

In this section, we describe the building blocks for our OPTI and MPC constructions.

2.1 Public Key Encryption

We take the definition of public key encryption directly from the work of Damgård et al. [DRSY23b].

Definition 1 (Public-Key Encryption (PKE)). *A public-key encryption (PKE) scheme is a tuple of efficient algorithms* $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *defined as follows.*

$\mathsf{KGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$**:** *The probabilistic algorithm* KGen *takes as input the security parameter* $\lambda \in \mathbb{N}$*, and outputs a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$*.*
$\mathsf{Enc}(\mathsf{pk}, \mathsf{msg}; \rho) \to ct$**:** *The probabilistic algorithm* Enc *takes as input the public key* pk*, a message* msg*, and implicit randomness* ρ*, and outputs a ciphertext* $ct = \mathsf{Enc}(\mathsf{pk}, \mathsf{msg}; \rho)$*.*
$\mathsf{Dec}(\mathsf{sk}, ct) \to \mathsf{msg}$**:** *The deterministic algorithm* Dec *takes as input the secret key* sk *and a ciphertext* ct *and outputs* $\mathsf{msg} = \mathsf{Dec}(\mathsf{sk}, ct)$ *which is either equal to some message* msg *or to an error symbol* $\perp$*.*

We require the following properties of a PKE scheme:

Correctness. We say that PKE *satisfies* correctness *if for all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$,

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{msg})) = \mathsf{msg}] = 1$$

(where the randomness is taken over the internal coin tosses of algorithm Enc*).*

CPA Security. We say that PKE *satisfies* semantic security *if for all PPT adversaries* $\mathcal{A}$*, for* $(\mathsf{msg}_0, \mathsf{msg}_1) \leftarrow \mathcal{A}(1^\lambda)$*, if* $|\mathsf{msg}_0| = |\mathsf{msg}_1|$*,*

$$\Pr\left[\mathcal{A}(\mathsf{pk}, ct) = b : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda); b \leftarrow \{0,1\}; \\ ct \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{msg}_b) \end{array}\right] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

(where the randomness is taken over the internal coin tosses of $\mathcal{A}$*,* KGen *and* Enc*).*

Key Verifiability. We say that PKE *satisfies* key verifiability *if there exists a deterministic polynomial-time algorithm* KeyVer *such that:*

- *For all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$,

$$\Pr[\mathsf{KeyVer}(\mathsf{pk}, \mathsf{sk}) = 1] = 1.$$

- *For all* pk, sk *such that* $(\mathsf{pk}, \mathsf{sk})$ *were not generated by* $\mathsf{KGen}(1^\lambda)$,

$$\Pr[\mathsf{KeyVer}(\mathsf{pk}, \mathsf{sk}) = 1] \leq \mathrm{negl}(\lambda).$$

ElGamal Instantiation. In our OPTI construction, we instantiate PKE with ElGamal encryption [ElG84] for both the public input slots and the public-recovery ciphertexts.

Let G be the cyclic group of prime order q generated by g, and let $h : \mathbb{Z}_q \to G$ be the efficiently invertible message encoding defined in Section 2.2.

The key-generation algorithm samples

$$\mathsf{sk} \leftarrow \mathbb{Z}_q \qquad \text{and sets} \qquad \mathsf{pk} := g^{\mathsf{sk}}.$$

For $\mathsf{msg} \in \mathbb{Z}_q$ and fresh randomness $\rho \leftarrow \mathbb{Z}_q$, encryption is defined as

$$ct \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{msg}; \rho) := (c_1, c_2) = \big(g^\rho, \mathsf{pk}^\rho \cdot h(\mathsf{msg})\big).$$

Given $ct = (c_1, c_2)$, decryption is defined as

$$\mathsf{Dec}(\mathsf{sk}, ct) := h^{-1}\big(c_2 \cdot c_1^{-\mathsf{sk}}\big).$$

Indeed,

$$c_2 \cdot c_1^{-\mathsf{sk}} = \mathsf{pk}^\rho \cdot h(\mathsf{msg}) \cdot \big(g^\rho\big)^{-\mathsf{sk}} = h(\mathsf{msg}).$$

The scheme is CPA-secure under the DDH assumption. Whenever the sender encrypts its two inputs in an execution of OPTI, the two encryption randomness values are sampled independently. The public-recovery mechanism only requires the general properties stated above; we instantiate it with the same ElGamal scheme. For this instantiation, key verifiability is obtained by setting

$$\mathsf{KeyVer}(\mathsf{pk}, \mathsf{sk}) = 1 \quad \Longleftrightarrow \quad \mathsf{pk} = g^{\mathsf{sk}}.$$

2.2 Commitment Scheme

We take the definition of a commitment scheme from the work of Damgård *et al* [DRSY23a].

A commitment scheme allows a party to commit to a value while keeping it hidden, and to later reveal the committed value with the guarantee that the commitment is binding [Ped92].

Definition 2 (Commitment Scheme). *A commitment scheme is a tuple of efficient algorithms* $\mathsf{COM} = (\mathsf{Com}, \mathsf{Decom})$ *defined as follows.*

$\mathsf{Com}(\mathsf{msg}) \to (o, c)$**:** *The commitment algorithm* Com *takes as input a message* $\mathsf{msg} \in \{0,1\}^\lambda$ *and outputs a decommitment value* $o \in \{0,1\}^\lambda$ *and a commitment value* $c \in \{0,1\}^\lambda$*.*
$\mathsf{Decom}(o, c) \to \{\mathsf{msg}, \bot\}$**:** *The opening algorithm* Decom *takes as input a decommitment value* $o \in \{0,1\}^\lambda$ *and a commitment value* $c \in \{0,1\}^\lambda$ *and outputs either a message* msg *or* $\bot$ *in case* o *is not a valid decommitment for* c*.*

— *We require the following properties of a commitment scheme:*

Correctness. We say that a commitment scheme COM *satisfies* correctness *if for any message* $\mathsf{msg} \in \{0,1\}^\lambda$ *and* $(o, c) \leftarrow \mathsf{Com}(\mathsf{msg})$*, we have that*

$$\Pr[\mathsf{Decom}(c, o) = \mathsf{msg}] \geq 1 - \mathrm{negl}(\lambda).$$

Hiding. The hiding property requires that a commitment reveals nothing about the underlying message. More formally, we say that a commitment scheme COM *satisfies* hiding *if for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr\left[b = b' \;\middle|\; \begin{array}{l} (\mathsf{msg}_0, \mathsf{msg}_1) \leftarrow \mathcal{A}(1^\lambda); \\ b \leftarrow \{0,1\}; \\ (o, c) \leftarrow \mathsf{Com}(\mathsf{msg}_b); \\ b' \leftarrow \mathcal{A}(c) \end{array} \right] \leq \frac{1}{2} + \mathrm{negl}(\lambda)\ .$$

Binding. The binding property requires that an adversary cannot open a commitment in two different ways. More formally, we say that a commitment scheme COM *satisfies* binding *if for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr\left[\mathsf{msg}_0, \mathsf{msg}_1 \neq \bot, \mathsf{msg}_0 \neq \mathsf{msg}_1 \;\middle|\; \begin{array}{l} (c, o_0, o_1) \leftarrow \mathcal{A}(1^\lambda); \\ \mathsf{msg}_0 \leftarrow \mathsf{Decom}(c, o_0); \\ \mathsf{msg}_1 \leftarrow \mathsf{Decom}(c, o_1) \end{array} \right] \leq \mathrm{negl}(\lambda)\ .$$

Instantiation: ElGamal Commitment We use a commitment scheme based on ElGamal encryption [ElG84]. Let G be a cyclic group of prime order q generated by g, where $|q| = \lambda$. We assume that the group parameters (G, q, g) are publicly known.

The key generation algorithm samples a secret key $\mathsf{sk} \leftarrow \mathbb{Z}_q$ and sets the public key $\mathsf{pk} := g^{\mathsf{sk}}$.

To commit to a message m, the committer samples randomness $r \leftarrow \mathbb{Z}_q$ and computes

$$\mathsf{Com}(m; r) := (c_1, c_2) = (g^r,\ \mathsf{pk}^r \cdot h(m)),$$

where $h : \mathbb{Z}_q \to G$ is an encoding of the message m into the group G. The opening of the commitment is (m, r).

The scheme is perfectly binding and computationally hiding under the Decisional Diffie–Hellman (DDH) assumption in G.

Message Encoding. To encode messages as group elements, we use the following standard embedding into a prime-order subgroup of $\mathbb{Z}_p^*$. Let $p = 2q + 1$ be a safe prime and let $G \subseteq \mathbb{Z}_p^*$ denote the subgroup of quadratic residues of order q. Given $m \in \mathbb{Z}_q$, define $h : \mathbb{Z}_q \to G$ by

$$h(m) := \begin{cases} m + 1, & \text{if } (m+1)^q \equiv 1 \pmod{p}, \\ -(m+1) \bmod p, & \text{otherwise.} \end{cases}$$

Decoding $M \in G$ is performed by interpreting M as an integer in $\mathbb{Z}_p$ and outputting

$$m := \begin{cases} M - 1, & \text{if } M \leq q, \\ p - M - 1, & \text{otherwise.} \end{cases}$$

By Euler's criterion, $(m+1)^q \equiv 1 \pmod{p}$ holds if and only if $m + 1$ is a quadratic residue modulo p, which guarantees that $h(m) \in G$. The mapping h is efficiently computable and invertible.

2.3 Secret Sharing

Definition 3 (Shamir secret sharing). *Let $\mathbb{F}$ be a finite field, and let $a_1, \ldots, a_n \in \mathbb{F}$ be distinct nonzero public evaluation points. For a reconstruction threshold r, Shamir secret sharing consists of the following algorithms:*

- $(s_1, \ldots, s_n) \leftarrow \mathtt{share}(s, n, r; \rho)$ *samples a uniformly random polynomial*

$$F(X) = s + \sum_{j=1}^{r-1} f_j X^j$$

of degree at most $r - 1$, where the coefficients $f_1, \ldots, f_{r-1}$ are determined by ρ, and outputs

$$s_i = F(a_i)$$

for every $i \in [n]$.

– $s \leftarrow \texttt{reconstruct}(s_{i_1}, \ldots, s_{i_r})$ *interpolates the unique polynomial of degree at most* $r-1$ *through the* r *given evaluations and outputs its constant coefficient* $F(0) = s$.

Shamir secret sharing satisfies the following properties:

Correctness: *For every* $s \in \mathbb{F}$ *and every* $T = \{i_1, \ldots, i_r\} \subseteq [n]$,

$$\Pr\left[\texttt{reconstruct}(s_{i_1}, \ldots, s_{i_r}) = s \mid (s_1, \ldots, s_n) \leftarrow \texttt{share}(s, n, r)\right] = 1.$$

Perfect Security: *For every* $s, s' \in \mathbb{F}$ *and every* $I \subseteq [n]$ *with* $|I| < r$, *the distributions*

$$\{(s_i)_{i \in I} \mid (s_1, \ldots, s_n) \leftarrow \texttt{share}(s, n, r)\}$$

and

$$\{(s'_i)_{i \in I} \mid (s'_1, \ldots, s'_n) \leftarrow \texttt{share}(s', n, r)\}$$

are identical.

Thus, a degree-d sharing is generated as $\texttt{share}(s, n, d+1)$. In the OPTI construction, we use $\mathbb{F} = \mathbb{Z}_q$, where q is the order of the group $\mathbb{G} = \langle g \rangle$ used for the receiver's coefficient commitments.

2.4 Multi-string Non-Interactive Zero-knowledge Proofs

We follow the definition of a multi-string NIZK (mNIZK) from [KRY24] and restate it here for completeness. We refer to [KRY24] for further details.

Definition 4 (multi-string NIZK scheme). *A* mNIZK *scheme for an NP relation* $\mathcal{R}_{\mathcal{L}}$ *has the following algorithms:*

$\mathsf{mNIZK.Gen}(1^\lambda) \to \texttt{crs}$ *: An algorithm to generate a common reference string. In the multi-string model comprising of* n *common reference strings, we let* $\overline{\texttt{crs}} = (\texttt{crs}_1, \ldots, \texttt{crs}_n)$ *denote the vector of the* n *common reference strings.*
$\mathsf{Prove}(\overline{\texttt{crs}}, \phi, w) \to \pi$**:** *An algorithm run by the prover that, given the vector of common reference strings* $\overline{\texttt{crs}}$, *statement* ϕ *and the witness* w *outputs the proof* π *that* $(\phi, w) \in \mathcal{R}_{\mathcal{L}}$.
$\mathsf{Verify}(\overline{\texttt{crs}}, \phi, \pi) \to \texttt{accept}/\texttt{reject}$**:** *An algorithm that, given the vector of common reference strings* $\overline{\texttt{crs}}$, *statement* ϕ *and the proof* π *verifies whether* π *proves the existence of a witness* w *such that* $(\phi, w) \in \mathcal{R}_{\mathcal{L}}$.

The rest of the algorithms are only necessary for proofs of security, and will not be used in the real world:

$\mathsf{S}_1(1^\lambda) \to (\texttt{crs}, \tau)$**:** *A simulation algorithm that generates a simulated reference string and a simulation trapdoor.*
$\mathsf{S}_2(\overline{\texttt{crs}}, \phi, \overline{\tau}) \to \pi$**:** *A simulation algorithm that, given the vector of common reference strings* $\overline{\texttt{crs}}$, *statement* ϕ *and a vector* $\overline{\tau}$ *containing* t_z *(where* t_z *is a pre-defined threshold for the multi-string NIZK proof system) simulation trapdoors for common reference strings in* $\overline{\texttt{crs}}$, *outputs a simulated proof of the existence of a witness* w *such that* $(\phi, w) \in \mathcal{R}_{\mathcal{L}}$.

$\mathsf{E}_1(1^\lambda) \to (\mathtt{crs}, \xi)$**:** *A simulation algorithm that generates a simulated reference string and an extraction trapdoor.*
$\mathsf{E}_2(\overline{\mathtt{crs}}, \phi, \pi, \overline{\xi}) \to w$**:** *An extraction algorithm that, given the vector of common reference strings $\overline{crs}$, statement ϕ, a valid proof π and a vector $\overline{\xi}$ containing t_s (where t_s is a pre-defined threshold for the multi-string NIZK proof system) extraction trapdoors for common reference strings in $\overline{crs}$, outputs a witness w such that $(\phi, w) \in \mathcal{R}_\mathcal{L}$.*
$\mathsf{SE}_1(1^\lambda) \to (\mathtt{crs}, \tau, \xi)$**:** *A simulation algorithm that outputs a simulated reference string, a simulation trapdoor and an extraction trapdoor such that $(\mathtt{crs}, \tau)$ is distributed as the output of S_1, and $(\mathtt{crs}, \xi)$ is distributed as the output of E_1.*

Properties We require the following properties from a (t_c, t_s, t_z, n) multi-string NIZK proof system for an NP relation $\mathcal{R}_\mathcal{L}$ (as defined in the work [GO07]).

(t_c, t_s, t_z, n)**-Completeness.** *Informally, this property requires that if at least t_c out of n common reference strings are honest, then the prover holding a witness for the statement should be able to create a convincing proof. More formally, for all non-uniform polynomial time adversaries $\mathcal{A}$,*

$$\Pr\left[\mathsf{Verify}(\overline{\mathtt{crs}}, \phi, \pi) = 1 \;\middle|\; \begin{array}{l}(\overline{\mathtt{crs}}, \phi, w) \leftarrow \mathcal{A}^{\mathsf{mNIZK.Gen}}(1^\lambda) \\ \pi \leftarrow \mathsf{Prove}(\overline{\mathtt{crs}}, \phi, w)\end{array}\right] \geq 1 - \mathrm{negl}(\lambda)$$

where mNIZK.Gen *on query i output $\mathtt{crs}_i \leftarrow \mathsf{mNIZK.Gen}(1^\lambda)$, at least t_c of the $\mathtt{crs}_i$'s generated by* mNIZK.Gen *are included and $\mathcal{A}$ outputs $(\phi, w) \in \mathcal{R}_\mathcal{L}$, and* negl *is a negligible function in the security parameter λ.*
We use multi-string NIZKs with perfect (t_c, t_s, t_z, n)-completeness for all $0 \leq t_c \leq n$ in our protocol. This means that even if the adversary chooses all common reference strings itself, we are guaranteed to output an acceptable proof when $(\phi, w) \in \mathcal{R}_\mathcal{L}$.

(t_c, t_s, t_z, n)**-Soundness.** *Informally, this property requires that if at least t_s out of n common random strings are honestly generated, then an adversary cannot forge the proof. The adversary gets to see possible choices of correctly generated common reference strings and can adaptively choose n of them. It may also include up to $n - t_s$ fake common reference strings it itself chooses. More formally, for all adversaries $\mathcal{A}$,*

$$\Pr[\mathsf{Verify}(\overline{\mathtt{crs}}, \phi, \pi) = 1 \text{ and } \phi \notin \mathcal{L} : (\overline{\mathtt{crs}}, \phi, \pi) \leftarrow \mathcal{A}^{\mathsf{mNIZK.Gen}}(1^\lambda)] \leq \mathrm{negl}(\lambda)$$

where mNIZK.Gen *is an oracle that on query i outputs $\mathtt{crs}_i \leftarrow \mathsf{mNIZK.Gen}(1^\lambda)$, the adversary outputs $\overline{\mathtt{crs}}$ such that at least t_s of the $\mathtt{crs}_i$'s generated by* mNIZK.Gen *are included, and* negl *is a negligible function in the security parameter λ.*

(t_c, t_s, t_z, n)**-Zero Knowledge.** *Informally, this property requires that if t_z common reference strings are correctly generated, then the adversary learns nothing from the proof. As is standard in the zero-knowledge literature, we say that this is the case when the proof can be simulated given only the statement ϕ.*
The definition of zero-knowledge is split into the following two parts.

Reference String Indistinguishability *This property simply says that the adversary cannot distinguish real common reference strings from simulated reference strings. More formally, for all non-uniform polynomial time adversaries $\mathcal{A}$,*

$$\left| \begin{array}{l} \Pr[\mathcal{A}(\mathtt{crs}) = 1 | \mathtt{crs} \leftarrow \mathsf{mNIZK.Gen}(1^\lambda)] \\ \\ -\Pr[\mathcal{A}(\mathtt{crs}) = 1 | (\mathtt{crs}, \tau) \leftarrow \mathsf{S}_1(1^\lambda)] \end{array} \right| \leq \mathrm{negl}(\lambda)$$

for a negligible function negl *in the security parameter λ.*

(t_c, t_s, t_z, n)*-Simulation Indistinguishability* *This property strengthens the standard definition of zero-knowledge and requires that even with access to the simulation trapdoors, the adversary cannot distinguish real proofs from simulated ones on a set of simulated reference strings. More formally, for all non-uniform polynomial time adversaries $\mathcal{A}$,*

$$\left| \begin{array}{l} \Pr[\mathcal{A}(\pi) = 1 | (\overline{\mathtt{crs}}, \overline{\tau}, \phi, w) \leftarrow \mathcal{A}^{\mathsf{S}_1}(1^\lambda); \pi \leftarrow \mathsf{Prove}(\overline{\mathtt{crs}}, \phi, w)] \\ \\ -\Pr[\mathcal{A}(\pi) = 1 : (\overline{\mathtt{crs}}, \overline{\tau}, \phi, w) \leftarrow \mathcal{A}^{\mathsf{S}_1}(1^\lambda); \pi \leftarrow \mathsf{S}_2(\overline{\mathtt{crs}}, \overline{\tau}, \phi)] \end{array} \right| \leq \mathrm{negl}(\lambda),$$

where S_1 on query i outputs $(\mathtt{crs}_i, \tau_i) \leftarrow \mathsf{S}_1(1^\lambda)$, the adversary outputs $(\phi, w) \in \mathcal{R}_\mathcal{L}$ and $\overline{\mathtt{crs}}, \overline{\tau}$ such that at least t_z of the $\mathtt{crs}_i$'s generated by S_1 are included and $\overline{\tau}$ contains t_z simulation trapdoors τ_i corresponding to $\mathtt{crs}_i$'s that have been generated by the oracle S_1, and negl *is a negligible function in the security parameter λ.*

(t_c, t_s, t_z, n)**-Knowledge.** *Informally, this property requires the existence of probabilistic polynomial time algorithms E_1 and E_2 that can extract a witness from a valid proof.*
Like the definition of zero-knowledge, the definition is split into two parts.

Reference String Indistinguishability *For all non-uniform polynomial time adversaries $\mathcal{A}$,*

$$\left| \begin{array}{l} \Pr[\mathcal{A}(\mathtt{crs}) = 1 | \mathtt{crs} \leftarrow \mathsf{mNIZK.Gen}(1^\lambda)] \\ \\ -\Pr[\mathcal{A}(\mathtt{crs}) = 1 | (\mathtt{crs}, \xi) \leftarrow \mathsf{E}_1(1^\lambda)] \end{array} \right| \leq \mathrm{negl}(\lambda)$$

where negl *is a negligible function in the security parameter λ.*

Extractability For all non-uniform polynomial time adversaries $\mathcal{A}$,

$$\Pr\left[\mathsf{Verify}(\overline{\boldsymbol{crs}},\phi,\pi)=1,(\phi,w)\notin\mathcal{R}_{\mathcal{L}}\middle|\begin{array}{c}(\overline{\boldsymbol{crs}},\phi,\pi)\leftarrow\mathcal{A}^{\mathsf{E}_1}(1^\lambda)\\ w\leftarrow\mathsf{E}_2(\overline{\boldsymbol{crs}},\phi,w,\overline{\xi})\end{array}\right]\leq\mathrm{negl}(\lambda)$$

where E_1 is an oracle that returns $(\boldsymbol{crs}_i,\xi_i)\leftarrow\mathsf{E}_1(1^\lambda)$, $\overline{\xi}$ contains at least t_s ξ_i's corresponding to the $\boldsymbol{crs}_i$'s generated by E_1, and negl *is a negligible function in the security parameter λ.*

(t_c,t_s,t_z,n)-Simulation Soundness. *Informally, this property requires that an adversary cannot prove any false statement even after seeing simulated proofs of arbitrary statements. More formally, for all non-uniform polynomial time adversaries $\mathcal{A}$, the following probability is negligible in the security parameter λ.*

$$\Pr[\mathsf{Verify}(\overline{\boldsymbol{crs}},\phi,\pi)=1,(\overline{\boldsymbol{crs}},\phi,\pi)\notin Q,\phi\notin\mathcal{L}|(\overline{\boldsymbol{crs}},\phi,\pi)\leftarrow\mathcal{A}^{\mathsf{S}_1,\mathsf{S}_2}(1^\lambda)]$$

where S_1 on query i returns $(\boldsymbol{crs}_i,\tau_i)\leftarrow\mathsf{S}_1(1^\lambda)$, S_2 on query $(\overline{\boldsymbol{crs}}_j,\phi_j)$ returns $\pi_j\leftarrow\mathsf{S}_2(\overline{\boldsymbol{crs}}_j,\overline{\tau}_j,\phi_j)$ with $\overline{\tau}_j$ having simulation trapdoors for the $\boldsymbol{crs}_i$'s generated by S_1, the adversary produces $\overline{\boldsymbol{crs}}_j$ containing at least t_s $\boldsymbol{crs}_i$'s generated by S_1, Q is the list of statements and corresponding proofs $(\overline{\boldsymbol{crs}}_j,\phi_j,\pi_j)$ in the queries to S_2.

(t_c,t_s,t_z,n)-Simulation Extractability. *Informally, this property requires that even after seeing many simulated proofs, whenever the adversary makes a new proof, we should be able to extract a witness. More formally, a multi-string NIZK proof system is (t_c,t_s,t_z,n)-simulation extractable if it has (t_c,t_s,t_z,n)-knowledge, is a (t_c,t_s,t_z,n)-NIZK proof (i.e. completeness, soundness and zero-knowledge hold for the relevant thresholds), and for all non-uniform polynomial time adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\overline{\boldsymbol{crs}},\phi,\pi)=1,\\ (\overline{\boldsymbol{crs}},\phi,\pi)\notin Q,\\ (\phi,w)\notin\mathcal{R}_{\mathcal{L}}\end{array}\middle|\begin{array}{c}(\overline{\boldsymbol{crs}},\phi,\pi)\leftarrow\mathcal{A}^{\mathsf{SE}_1',\mathsf{S}_2}(1^\lambda),\\ w\leftarrow\mathsf{E}_2(\overline{\boldsymbol{crs}},\phi,\pi,\overline{\xi})\end{array}\right]\leq\mathrm{negl}(\lambda)$$

where SE_1' on query i returns $(\boldsymbol{crs}_i,\xi_i)$ from $(\boldsymbol{crs}_i,\tau_i,\xi_i)\leftarrow\mathsf{SE}_1(1^\lambda)$, S_2 on query $(\overline{\boldsymbol{crs}}_j,\phi_j)$ returns $\pi_j\leftarrow\mathsf{S}_2(\overline{\boldsymbol{crs}}_j,\overline{\tau}_j,\phi_j)$ (where $\overline{\tau}_j$ contains t_z τ_i's corresponding to $\boldsymbol{crs}_i$'s in $\overline{\boldsymbol{crs}}_j$ generated by SE_1), Q is the list of statements and corresponding proofs $(\overline{\boldsymbol{crs}}_j,\phi_j,\pi_j)$ made by S_2, $\overline{\xi}$ contains the first t_s ξ_i's generated by SE_1 corresponding to $\boldsymbol{crs}_i$'s in $\overline{\boldsymbol{crs}}$, and negl *is a negligible function in the security parameter λ.*

The above property of simulation extractability implies simulation-soundness.

Instantiation In our constructions, we use the $(0, t_s, t_z, n)$ multi-string NIZK with $t_s = t_z = \lceil (n+1)/2 \rceil$ of [GO07], which relies on enhanced trapdoor permutations and satisfies the properties outlined above.

2.5 Garbling Scheme

Definition 5. *[DRSY23b] A* garbling scheme with independent key generation *$\mathcal{G}$ is a pair of algorithms* $(\mathsf{KGen}, \mathtt{Garble}, \mathsf{Eval})$ *defined as follows.*

- $\mathbf{K} \leftarrow \mathsf{KGen}(1^\lambda, \mathsf{L}, \rho^{\mathtt{GC}})$*: The algorithm* KGen *takes as input a unary representation of the security parameter λ and a number of input wires and outputs L pairs of garbled labels $\mathbf{K} = (K_0^1, K_1^1, \ldots, K_0^{\mathsf{L}}, K_1^{\mathsf{L}})$. For simplicity, we assume that for every $i \in [\mathsf{L}]$ and $b \in \{0,1\}$, it holds that $K_b^i \in \{0,1\}^\lambda$.*
- $\mathtt{GC} \xleftarrow{\$} \mathtt{Garble}(1^\lambda, C, \mathbf{K}, \rho^{\mathtt{GC}})$*: The algorithm* $\mathtt{Garble}$ *takes as input a unary representation of the security parameter λ, a Boolean circuit $C : \{0,1\}^{\mathsf{L}} \to \{0,1\}^m$ and keys $\mathbf{K}$, and outputs a garbled circuit* $\mathtt{GC}$*.*
- $y \leftarrow \mathsf{Eval}(\mathtt{GC}, K^1, \ldots, K^{\mathsf{L}})$*: The algorithm* Eval *takes as input the garbled circuit* $\mathtt{GC}$ *and* L *garbled labels $K^1, \ldots, K^{\mathsf{L}}$, and outputs a value $y \in \{0,1\}^m$.*

We require the following properties of a garbling scheme:

- **Perfect Correctness**. We say $\mathcal{G}$ satisfies *perfect correctness* if for any Boolean circuit $C : \{0,1\}^{\mathsf{L}} \to \{0,1\}^m$ and $x = (x_1, \ldots, x_{\mathsf{L}})$ it holds that

$$\Pr\big[\mathsf{Eval}(\mathtt{GC}, \mathbf{K}[x]) = C(x)\big] = 1,$$

  where $\mathbf{K} \leftarrow \mathsf{KGen}(1^\lambda, \mathsf{L}, \rho^{\mathsf{KGen}})$ with $\mathbf{K} = (K^{0,1}, K^{1,1}, \ldots, K^{0,\mathsf{L}}, K^{1,\mathsf{L}})$, $\mathtt{GC} \leftarrow \mathtt{Garble}(1^\lambda, C, \mathbf{K}, \rho^{\mathtt{GC}})$, and $\mathbf{K}[x] := (K^{x_1,1}, \ldots, K^{x_{\mathsf{L}},\mathsf{L}})$.
- **Privacy.** We say that $\mathcal{G}$ satisfies *privacy* if there exists a simulator $\mathtt{simGC}$ such that for every PPT adversary $\mathcal{A}$, it holds that

$$\Pr[\mathtt{GC}^{\mathtt{priv}} = 1] \leq \frac{1}{2} + \mathrm{negl}(\lambda),$$

  where $\mathtt{GC}^{\mathtt{priv}}$ experiment is defined in the following figure:

The garbling scheme privacy experiment $\mathtt{GC}^{\mathtt{priv}}$.

1. $\mathcal{A}$ constructs a circuit $C : \{0,1\}^{\mathsf{L}} \to \{0,1\}^m$, outputs an input $x = (x_1, \ldots, x_{\mathsf{L}}) \in \{0,1\}^{\mathsf{L}}$ and sends them to the challenger.
2. The challenger chooses a random bit $b \in \{0,1\}$ and does the following:
   (a) If $b = 0$:
      - Sample labels $\mathbf{K} \leftarrow \mathsf{KGen}(1^\lambda, \mathsf{L}, \rho^{\mathsf{KGen}})$.
      - Sample garbled circuit $\mathtt{GC} \leftarrow \mathtt{Garble}(1^\lambda, C, \mathbf{K}, \rho^{\mathtt{GC}})$.
      - Set $K^i := K^{x_i,i}$ for $i \in [\mathsf{L}]$.

   (b) If $b = 1$:
      - Sample labels $\mathbf{K} \leftarrow \mathsf{KGen}(1^\lambda, \mathsf{L}, \rho^{\mathsf{KGen}})$, s.t. only one label is sampled for every wire.

– Sample simulated garbling $\mathtt{GC} \leftarrow \mathtt{simGC}(C, C(x), \mathbf{K}, \rho^{\mathtt{GC}})$.
– Set $K^i := K^{x_i,i}$ for $i \in [\mathsf{L}]$.

3. The challenger sends $\mathtt{GC}, K^1, \ldots, K^{\mathsf{L}}$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a bit b'.
5. $\mathcal{A}$ wins if $b = b'$, and we say that the experiment evaluates to 1 in this case.

3 Oblivious Public Transfer with Intermediaries (OPTI)

In this section, we introduce a new building block called Oblivious Public Transfer with Intermediaries (OPTI). As in standard Oblivious Transfer (OT), a sender S holds a pair of messages (x_0, x_1) and a receiver R holds a choice bit b, with the transferred value being x_b. The key difference is that, in OPTI, the output x_b is public and delivered to all participating parties, referred to as intermediaries.

Although the public output allows S to infer R's choice whenever $x_0 \neq x_1$, OPTI ensures *privacy* in the sense that no information about b is revealed to the adversary as long as both S and R are honest. In addition, if the transfer fails, meaning the parties do not obtain the value x_b, this failure can only be caused by corruption of either S or R, and *identifiability* guarantees that all parties are able to identify the corrupt party. Finally, as in standard OT, OPTI guarantees *sender-privacy*: if the sender is honest, the adversary learns nothing about x_{1-b}.

Looking ahead, OPTI enables the verifiable transfer of private messages (denoted as b above) from a sender to a receiver in our IA protocol, ensuring that the transfer either succeeds or results in unanimous cheater identification.

Below, we give a formal description of OPTI in Section 3.1 and present a 2-round construction realizing OPTI in Section 3.2.

3.1 OPTI Definitions

The OPTI protocol involves a sender S, a receiver R, and $n-2$ additional parties. We do not assign fixed indices for S and R, as different parties may assume these roles. The algorithm executed by party $\mathcal{P}_k$ in Round r ($r \in [2]$) is denoted OPTI_r^k, while the output computation algorithm is denoted by $\mathsf{OPTI}_{\mathsf{output}}$ (which is identical for all parties).

The purpose of the following definitions is to capture exactly the properties of OPTI that are needed by our compiler. Rather than giving a general simulation-based definition, we formulate these properties through separate security games, which allows us to isolate the guarantees required from OPTI without imposing a stronger security notion than necessary.

Definition 6 (Oblivious Public Transfer with Intermediaries – OPTI). *A two-round 1-out-of-2 oblivious public transfer with intermediaries* $\mathsf{OPTI} =$

$\left((\mathsf{OPTI}_1^{\mathsf{R}}, \mathsf{OPTI}_2^{\mathsf{R}}), (\mathsf{OPTI}_1^{\mathsf{S}}, \mathsf{OPTI}_2^{\mathsf{S}}), ((\mathsf{OPTI}_1^{k}, \mathsf{OPTI}_2^{k})_{k\in[n]}), \mathsf{OPTI}_{\mathsf{output}}\right)$ *is an interactive protocol among n parties, where one party plays the role of the receiver* R*, one party plays the role of the sender* S*, and every party $k \in [n]$ (including* R *and* S*) plays the role of an intermediary party $\mathcal{P}_k$. The receiver* R *has input $b \in \{0,1\}$, the sender* S *has input (x_0, x_1), while intermediary parties $\mathcal{P}_k$ for $k \in [n] \setminus \{\mathsf{R}, \mathsf{S}\}$ have no input. At the end of the protocol, every party $\mathcal{P}_k$ outputs $\mathsf{OPTI}_{\mathsf{output}} = x_b$. We require that an OPTI protocol satisfies sender privacy (definition 7), privacy (definition 8), and identifiability (definition 9).*

Next, we present game-based definitions that formalize the security guarantees of OPTI, namely *sender-privacy*, *privacy* and *identifiability*. In the games below, we use msg_r^k and $\mathsf{msg}_r^{k\to l}$ to denote $\mathcal{P}_k$'s rth round (where $r \in [2]$) broadcast and point-to-point communication to $\mathcal{P}_l$ respectively.

Sender privacy. Informally, if the sender is honest, the adversary should learn nothing about the unchosen sender input beyond what is revealed by the public output.

Definition 7 (OPTI: Sender Privacy). *An OPTI scheme is sender-private if for security parameter $\lambda \in \mathbb{N}$, for every PPT adversary $\mathcal{A}$, it holds that $Pr[\mathcal{A}\ wins] \leq \frac{1}{2} + negl(\lambda)$ in the experiment in Figure 1.*

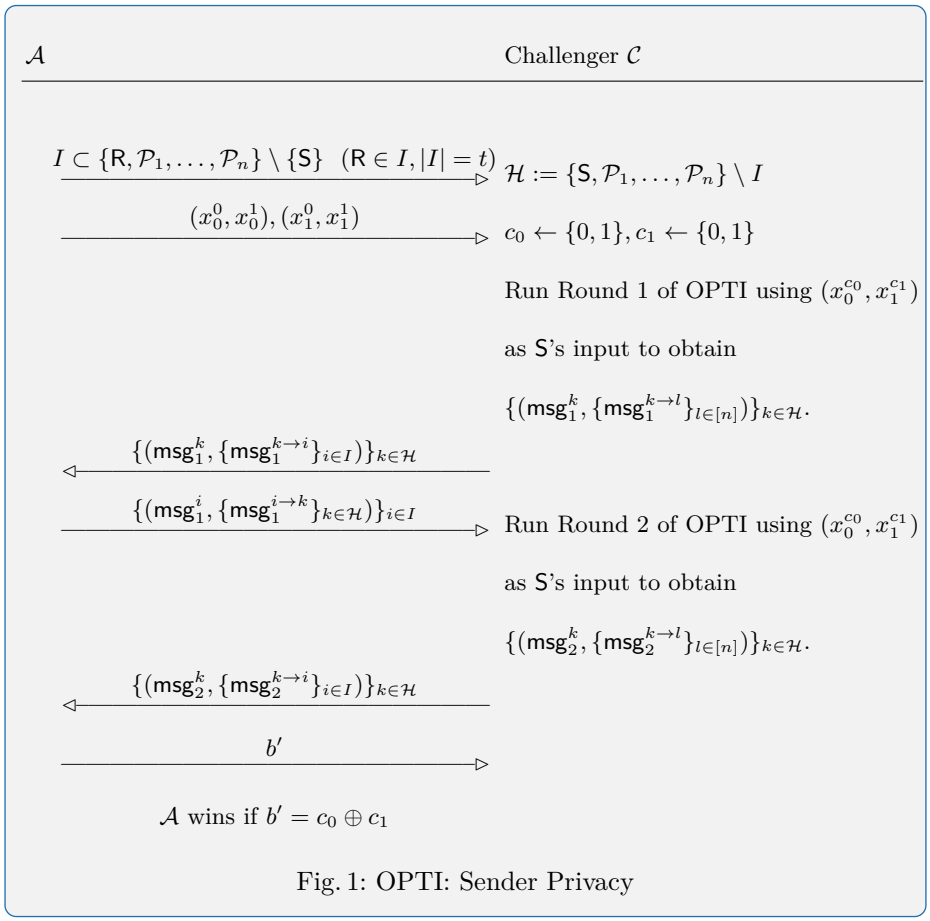

Fig. 1: OPTI: Sender Privacy

Privacy. Informally, if both the sender and receiver are honest, the adversary should learn nothing about the receiver's choice bit beyond what is revealed by the public output.

Definition 8 (OPTI: Privacy). *An OPTI scheme is private if for security parameter $\lambda \in \mathbb{N}$, for every PPT adversary $\mathcal{A}$, it holds that $Pr[\mathcal{A}\ wins] \leq \frac{1}{2} + negl(\lambda)$ in the experiment in Figure 2.*

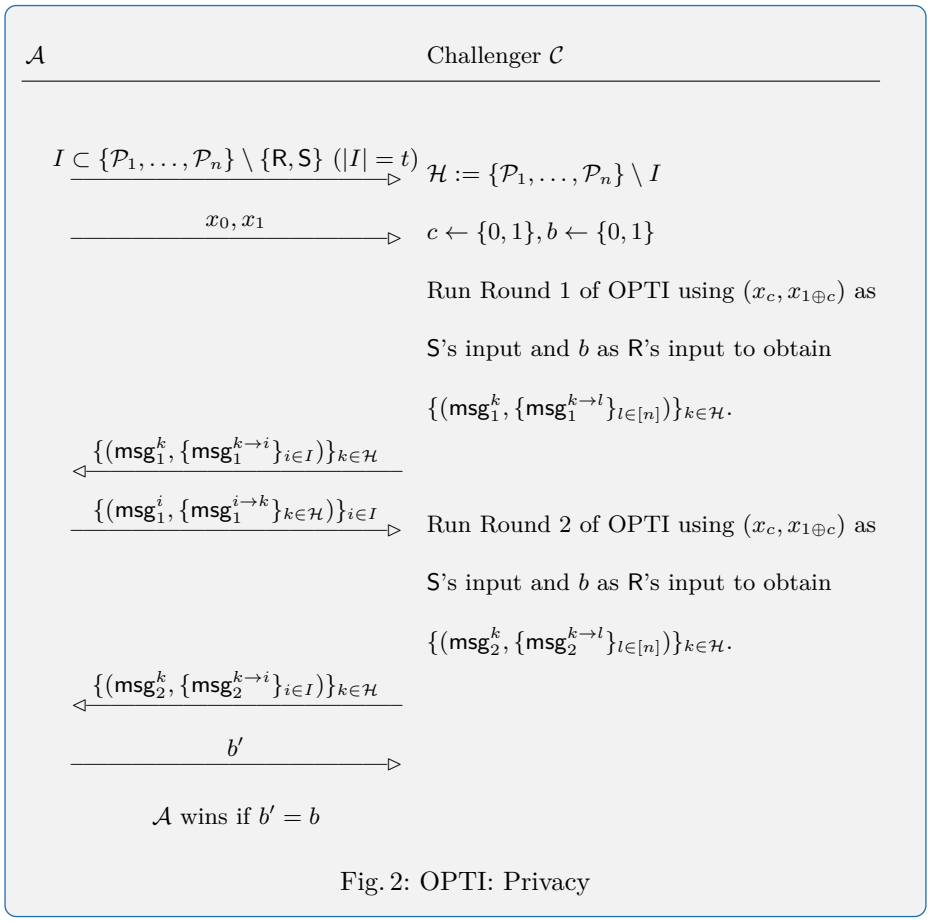

Fig. 2: OPTI: Privacy

Identifiability. Informally, honest parties should either obtain a valid and consistent output or identify a corrupt sender or receiver.

Definition 9 (OPTI: Identifiability). *An OPTI scheme achieves identifiability if for security parameter $\lambda \in \mathbb{N}$, for every PPT adversary $\mathcal{A}$, it holds that $Pr[\mathcal{A}$ wins$] \leq negl(\lambda)$ in the experiment in Figure 3.*

$\mathcal{A}$		Challenger $\mathcal{C}$
	$I \subset \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ $(\lvert I\rvert = t)$ $\longrightarrow$	$\mathcal{H} := \{\mathcal{P}_1, \ldots, \mathcal{P}_n\} \setminus I$.
	a) If $\mathsf{S}, \mathsf{R} \in \mathcal{H} : b, (x_0, x_1)$ $\longrightarrow$	Run Round 1 of OPTI using (x_0, x_1) as S's input and b as R's input to obtain $\{(\mathsf{msg}_1^k, \{\mathsf{msg}_1^{k\to l}\}_{l\in[n]})\}_{k\in\mathcal{H}}$.
	b) If $\mathsf{S} \in \mathcal{H}$ and $\mathsf{R} \in I$:(x_0, x_1) $\longrightarrow$	Run Round 1 of OPTI using (x_0, x_1) as S's input to obtain $\{(\mathsf{msg}_1^k, \{\mathsf{msg}_1^{k\to l}\}_{l\in[n]})\}_{k\in\mathcal{H}}$.
	c) If $\mathsf{S} \in I$ and $\mathsf{R} \in \mathcal{H} : b$ $\longrightarrow$	Run Round 1 of OPTI using b as R's input to obtain $\{(\mathsf{msg}_1^k, \{\mathsf{msg}_1^{k\to l}\}_{l\in[n]})\}_{k\in\mathcal{H}}$.
	d) If $\mathsf{S}, \mathsf{R} \in I : \emptyset$ $\longrightarrow$	Run Round 1 of OPTI to obtain $\{(\mathsf{msg}_1^k, \{\mathsf{msg}_1^{k\to l}\}_{l\in[n]})\}_{k\in\mathcal{H}}$.
	$\longleftarrow$ $\{(\mathsf{msg}_1^k, \{\mathsf{msg}_1^{k\to i}\}_{i\in I})\}_{k\in\mathcal{H}}$	
	$\longrightarrow$ $\{(\mathsf{msg}_1^i, \{\mathsf{msg}_1^{i\to k}\}_{k\in\mathcal{H}})\}_{i\in I}$	a) If $\mathsf{S}, \mathsf{R} \in \mathcal{H}$: Run Round 2 of OPTI using (x_0, x_1) as S's input and b as R's input to obtain $\{(\mathsf{msg}_2^k, \{\mathsf{msg}_2^{k\to l}\}_{l\in[n]})\}_{k\in\mathcal{H}}$. b) If $\mathsf{S} \in \mathcal{H}$ and $\mathsf{R} \in I$: Run Round 2 of OPTI using (x_0, x_1) as S's input to obtain $\{(\mathsf{msg}_2^k, \{\mathsf{msg}_2^{k\to l}\}_{l\in[n]})\}_{k\in\mathcal{H}}$. c) If $\mathsf{S} \in I$ and $\mathsf{R} \in \mathcal{H}$: Run Round 2 of OPTI using b as R's input to obtain $\{(\mathsf{msg}_2^k, \{\mathsf{msg}_2^{k\to l}\}_{l\in[n]})\}_{k\in\mathcal{H}}$.
	$\longleftarrow$ $\{(\mathsf{msg}_2^k, \{\mathsf{msg}_2^{k\to i}\}_{i\in I})\}_{k\in\mathcal{H}}$	d) If $\mathsf{S}, \mathsf{R} \in I$: Run Round 2 of OPTI to obtain $\{(\mathsf{msg}_2^k, \{\mathsf{msg}_2^{k\to l}\}_{l\in[n]})\}_{k\in\mathcal{H}}$.
	$\longrightarrow$ $\{(\mathsf{msg}_2^i, \{\mathsf{msg}_2^{i\to k}\}_{k\in\mathcal{H}})\}_{i\in I}$	

$\mathcal{A}$ wins if any of the following hold:

a) $\mathsf{S}, \mathsf{R} \in \mathcal{H}$ and $\exists \mathcal{P}_k \in \mathcal{H}$ s.t. $\mathsf{OPTI}^k_{\mathsf{output}} \in \{\mathsf{abort}, \mathsf{abort}_\mathsf{S}, \mathsf{abort}_\mathsf{R}, x\}$ for $x \neq x_b$,
b) $\mathsf{S} \in \mathcal{H}$ and $\mathsf{R} \in I$ and $\exists \mathcal{P}_k \in \mathcal{H}$ s.t. $\mathsf{OPTI}^k_{\mathsf{output}} \in \{\mathsf{abort}, \mathsf{abort}_\mathsf{S}, x\}$ for $x \notin \{x_0, x_1\}$,
c) $\mathsf{S} \in I$ and $\mathsf{R} \in \mathcal{H}$ and $\exists \mathcal{P}_k \in \mathcal{H}$ s.t. $\mathsf{OPTI}^k_{\mathsf{output}} \in \{\mathsf{abort}, \mathsf{abort}_\mathsf{R}\}$,
d) $\mathsf{S}, \mathsf{R} \in I$ and $\exists \mathcal{P}_k \in \mathcal{H}$ s.t. $\mathsf{OPTI}^k_{\mathsf{output}} = \mathsf{abort}$,
e) $\exists \mathcal{P}_k, \mathcal{P}_l \in \mathcal{H}$ s.t. $\mathsf{OPTI}^k_{\mathsf{output}} \neq \mathsf{OPTI}^l_{\mathsf{output}}$.

Fig. 3: OPTI: Identifiability

3.2 OPTI Protocol

In this section, we present a two-round construction that realizes OPTI. At a high level, in the first round the sender S samples two distinct random ElGamal secret keys $\mathsf{sk}^0, \mathsf{sk}^1$ and threshold-shares them with threshold $(t+1)$ among the intermediaries. It additionally distributes a threshold-$(2t+1)$ sharing of 0. The receiver R samples a degree-t polynomial $B(X)$ whose constant term is its input bit b and sends each intermediary $\mathcal{P}_k$ the evaluation $\beta_k = B(k)$.

The sender broadcasts commitments to x_0 and x_1, as well as commitments to its individual shares of sk^0, sk^1, and 0, and privately sends the corresponding shares and openings. The receiver instead broadcasts a coefficient transcript $\mathit{\Gamma}_{\mathsf{R}}$ that binds all the values β_k to evaluations of the same degree-t polynomial. Together with β_k, the receiver privately sends an auxiliary value ρ_k that enables $\mathcal{P}_k$ to verify this consistency locally.

Each intermediary $\mathcal{P}_k$ that receives valid data from both parties locally computes

$$\langle y \rangle_k = (1-\beta_k)[\mathsf{sk}^0]_k + \beta_k[\mathsf{sk}^1]_k + \langle 0 \rangle_k.$$

Here $[m]_k$ denotes a threshold-$(t+1)$ share of m, and $\langle m \rangle_k$ denotes a threshold-$(2t+1)$ share of m. The resulting shares reconstruct

$$y = (1-b)\mathsf{sk}^0 + b\mathsf{sk}^1.$$

In the second round, the sender also broadcasts encryptions of x_0 and x_1 under the public keys corresponding to sk^0 and sk^1, respectively, in a randomly permuted order. It proves in zero knowledge that these ciphertexts contain the committed inputs under the keys defined by the committed sharings, without revealing their order. If the receiver's proof is accepted, then $b \in \{0,1\}$ and hence $y = \mathsf{sk}^b$. The parties identify the unique ciphertext whose public key is g^y, decrypt it using y, and obtain x_b.

Several challenges arise in making this approach work. First, the products of the degree-t receiver and sender sharings produce a degree-$2t$ polynomial that could reveal information beyond its constant term. We therefore randomize it by adding an independently generated threshold-$(2t+1)$ sharing of 0. This ensures that the resulting polynomial reveals only $y = (1-b)\mathsf{sk}^0 + b\mathsf{sk}^1$ about the two shared keys.

The use of random keys is important because the receiver's proof that $B(0)$ is a bit is broadcast only in the second round. A corrupt receiver may therefore see the honest parties' second-round messages before its invalid proof is rejected. If $B(0) \notin \{0,1\}$, the receiver learns at most one affine relation between the two random secret keys. Under DDH, this relation does not allow it to decrypt either independently randomized sender ciphertext.

This indirection through random keys is essential for sender privacy. If the sender inputs (which are garbled-circuit labels in our application) were shared directly, then a receiver choosing $b^\star = B(0) \notin \{0,1\}$ would learn the input-dependent affine combination $(1-b^\star)x_0 + b^\star x_1$. This constitutes additional information about both sender inputs and, in general, violates sender privacy. By

sharing random keys instead, the affine relation concerns only input-independent randomness, while the sender inputs remain protected inside the ElGamal ciphertexts.

The invalid receiver is then identified when its bitness proof fails.

Second, reconstructing y requires at least $(2t + 1)$ valid intermediary contributions. As described above, an intermediary can produce its contribution directly only if it received valid key shares and openings from S and a receiver pair (β_k, ρ_k) that verifies against Γ_{R}.

When no cheater is identified, at least $n - t$ parties receive valid sender data, and at least $n - t$ parties receive a valid receiver evaluation pair. However, this guarantees only $n - 2t$ parties that received valid data from both. At the tight threshold $n = 3t + 1$, this may be only $t + 1$ parties, which is insufficient to reconstruct a degree-$2t$ polynomial.

To address this, if $\mathcal{P}_k$ did not receive valid data from S but did receive a valid pair (β_k, ρ_k) from R, then in the second round it broadcasts this pair together with the secret key corresponding to the recovery public key it broadcast in the first round. Simultaneously, S broadcasts encryptions of $[\mathsf{sk}^0]_k$, $[\mathsf{sk}^1]_k$, and $\langle 0 \rangle_k$ under that recovery public key.

Everyone first verifies the disclosed pair (β_k, ρ_k) against Γ_{R} and checks the disclosed recovery key. If these checks succeed, the sender's encrypted shares are decrypted and the same local computation described above is used to recover the missing contribution of $\mathcal{P}_k$.

After all contributions have been verified, the parties reconstruct y if at least $2t + 1$ valid contributions remain; otherwise, they identify R. This cannot incorrectly identify an honest receiver because, when R is honest, all at least $n - t \geq 2t + 1$ honest intermediaries provide either a valid direct contribution or a valid recovered contribution. This completes the high-level overview of our construction.

To summarize, our protocol proceeds as follows:

1. In the first round, S commits to its two inputs, distributes degree-t shares of two random secret keys and a degree-$2t$ sharing of 0, broadcasts commitments to these shares, and privately sends the corresponding shares and openings. The receiver broadcasts Γ_{R} and privately sends each $\mathcal{P}_k$ an evaluation pair (β_k, ρ_k). Each intermediary later verifies its pair against Γ_{R}. In addition, all intermediaries broadcast their recovery public keys and CRS strings.
2. In the second round, R proves that Γ_{R} represents a degree-t polynomial whose constant term is a bit. The sender broadcasts recovery encryptions of its key shares and zero-shares, as well as the two encrypted input slots and a proof linking them to the committed inputs and shared keys. Parties that received valid data from both S and R compute and broadcast their degree-$2t$ shares $\langle y \rangle_k$ together with proofs of correct computation. A party that received a valid receiver pair but no valid sender data broadcasts its recovery secret key and receiver pair.

3. To compute the output, the parties verify the disclosed receiver pairs, decrypt the corresponding sender shares, and compute any missing shares of y. They reconstruct y, identify the unique encrypted input slot whose public key equals g^y, and decrypt that slot to obtain x_b.

Informally, the above protocol design ensures that the protocol successfully computes the output whenever no party is identified as cheating because of the following reason: An adversary attempting to avoid detection can withhold shares from at most t honest parties on behalf of a corrupt S and at most t honest parties on behalf of a corrupt R. The most delicate situation arises when these two sets are disjoint. In that case, up to $2t$ (among $\geq 2t+1$) honest parties may be missing some shares, preventing them from locally computing their degree-$2t$ shares. However, the adversary gains no real advantage by dropping shares on behalf of S. This is because the protocol design forces the potentially corrupt S (if it wishes to avoid detection) to broadcast encrypted versions of all shares, and the degree-$2t$ shares of honest parties that received a valid receiver evaluation pair can still be reconstructed using these encryptions. Consequently, the only degree-$2t$ shares that may remain unavailable are those corresponding to parties unhappy with R – but since there can be at most only t such parties, this does not suffice to abort reconstruction. This completes a high-level overview of our construction.

Receiver coefficient transcript. Let $\mathbb{G}$ be a cyclic group of prime order q with public generator g, and instantiate Shamir sharing over $\mathbb{F} = \mathbb{Z}_q$. Let $a_1, \ldots, a_n \in \mathbb{F} \setminus \{0\}$ be the distinct public Shamir evaluation points.

The receiver represents its sharing and an independent auxiliary sharing by

$$B(X) = \sum_{j=0}^{t} b_j X^j, \qquad R_B(X) = \sum_{j=0}^{t} r_j X^j.$$

It samples $\alpha \leftarrow \mathbb{Z}_q^*$, sets $U = g^\alpha$, and broadcasts

$$\Gamma_{\mathsf{R}} = \left(U, \{(A_j, D_j)\}_{j=0}^{t}\right), \qquad A_j = g^{r_j}, \qquad D_j = U^{r_j} g^{b_j}.$$

For $\beta, \rho \in \mathbb{Z}_q$, define $\mathsf{ValidEval}(\Gamma_{\mathsf{R}}, k, \beta, \rho) = 1 \iff U, A_j, D_j \in \mathbb{G}\ \forall j$, $U \neq 1_{\mathbb{G}}$ and

$$g^\rho = \prod_{j=0}^{t} A_j^{k^j}, \qquad U^\rho g^\beta = \prod_{j=0}^{t} D_j^{k^j}.$$

Indeed, for an honestly generated transcript,

$$\prod_{j=0}^{t} A_j^{k^j} = g^{\sum_{j=0}^{t} r_j k^j} = g^{R_B(k)} \text{ and } \prod_{j=0}^{t} D_j^{k^j} = U^{R_B(k)} g^{B(k)}.$$

Because g has order q, the first equation fixes $\rho = R_B(k)$. Substituting this equality into the second equation and cancelling the U-term fixes $\beta = B(k)$. Thus every accepted pair is bound to the polynomial represented by Γ_{R}.

Relations for the mNIZKs. In our construction, we use multi-string NIZKs (Section 2.4) to enable parties to prove the correctness of their actions. In more detail, each party publishes its own CRS in the first round, and these strings are combined to form a joint CRS. As long as a majority of the individual CRS are generated honestly, proofs produced with respect to the joint CRS satisfy the standard zero-knowledge guarantees. The corresponding NIZK relations are defined below.

We utilize three relations: $\mathcal{R}_{\mathsf{send}}$, for proving that the sender's committed key sharings, recovery ciphertexts, and public input slots are mutually consistent; $\mathcal{R}_{\mathsf{rec}}$, for proving that the receiver's coefficient transcript is well-formed and has a bit as its constant term; and $\mathcal{R}_{\mathsf{comp}}$, for proving correct local computation using an evaluation bound to the receiver's coefficient transcript.

$$
\mathcal{R}_{\mathsf{send}} = \left\{ \begin{array}{l|l}
 & \mathsf{sk}^0, \mathsf{sk}^1 \in \mathbb{Z}_q,\ \mathsf{sk}^0 \neq \mathsf{sk}^1,\ \sigma \in \{0,1\}, \\
 & \forall c \in \{0,1\},\ \forall k \in [n] : \\
\phi_{\mathsf{send}} = \big(\{c^{x_c}\}_{c\in\{0,1\}}, & x_c = \mathsf{Decom}(o^{x_c}, c^{x_c}), \\
\quad \{(\{(c_k^{\mathsf{sk}^c}, ct_k^{\mathsf{sk}^c})\}_{c\in\{0,1\}}, & ([\mathsf{sk}^c]_1, \ldots, [\mathsf{sk}^c]_n) = \\
\quad\quad c_k^0, ct_k^0, \mathsf{pk}_k)\}_{k\in[n]}, & \mathtt{share}(\mathsf{sk}^c, n, t+1; \rho^{\mathsf{sk}^c}), \\
\quad \{(\mathsf{pk}_d^{\mathsf{slot}}, ct_d^{\mathsf{slot}})\}_{d\in\{0,1\}}\big), & (\langle 0\rangle_1, \ldots, \langle 0\rangle_n) = \\
w_{\mathsf{send}} = \big(\sigma, \rho^0, & \mathtt{share}(0, n, 2t+1; \rho^0), \\
\quad \{(x_c, o^{x_c}, \mathsf{sk}^c, & [\mathsf{sk}^c]_k = \mathsf{Decom}(o_k^{\mathsf{sk}^c}, c_k^{\mathsf{sk}^c}), \\
\quad\quad \rho^{\mathsf{sk}^c}, \rho_{\mathsf{slot}}^c)\}_{c\in\{0,1\}}, & \langle 0\rangle_k = \mathsf{Decom}(o_k^0, c_k^0), \\
\quad \{(\{(o_k^{\mathsf{sk}^c}, \rho_{\mathsf{enc},k}^{\mathsf{sk}^c})\}_{c\in\{0,1\}}, & ct_k^{\mathsf{sk}^c} = \mathsf{Enc}(\mathsf{pk}_k, [\mathsf{sk}^c]_k; \rho_{\mathsf{enc},k}^{\mathsf{sk}^c}), \\
\quad\quad o_k^0, \rho_{\mathsf{enc},k}^0)\}_{k\in[n]}\big) & ct_k^0 = \mathsf{Enc}(\mathsf{pk}_k, \langle 0\rangle_k; \rho_{\mathsf{enc},k}^0), \\
 & \mathsf{pk}_{c\oplus\sigma}^{\mathsf{slot}} = g^{\mathsf{sk}^c}, \\
 & ct_{c\oplus\sigma}^{\mathsf{slot}} = \mathsf{Enc}(\mathsf{pk}_{c\oplus\sigma}^{\mathsf{slot}}, x_c; \rho_{\mathsf{slot}}^c)
\end{array} \right\}.
$$

$$\mathcal{R}_{\mathsf{rec}} = \left\{ \begin{array}{l|ll} \phi = \Gamma_{\mathsf{R}} = \big(U, \{(A_j, D_j)\}_{j=0}^{t}\big) & \alpha \in \mathbb{Z}_q^* \ \wedge \ U = g^{\alpha} & \\ & \forall j \in \{0, \ldots, t\} : & b_j, r_j \in \mathbb{Z}_q \\ w = \big(\alpha, \{(b_j, r_j)\}_{j=0}^{t}\big) & \forall j \in \{0, \ldots, t\} : & A_j = g^{r_j} \\ & \forall j \in \{0, \ldots, t\} : & D_j = U^{r_j} g^{b_j} \\ & b_0(b_0 - 1) = 0 & \end{array} \right\}.$$

$$\mathcal{R}_{\mathsf{comp}} = \left\{ \begin{array}{l|l} \phi_{\mathsf{comp}} = \big(\langle y \rangle_k, \Gamma_{\mathsf{R}}, k, c_k^{\mathsf{sk}^0}, c_k^{\mathsf{sk}^1}, c_k^0\big), & \mathsf{ValidEval}(\Gamma_{\mathsf{R}}, k, \beta_k, \rho_k) = 1, \\ w_{\mathsf{comp}} = \big((\beta_k, \rho_k), ([\mathsf{sk}^0]_k, o_k^{\mathsf{sk}^0}), & [\mathsf{sk}^0]_k \leftarrow \mathsf{Decom}(o_k^{\mathsf{sk}^0}, c_k^{\mathsf{sk}^0}), \\ ([\mathsf{sk}^1]_k, o_k^{\mathsf{sk}^1}), (\langle 0 \rangle_k, o_k^0)\big) & [\mathsf{sk}^1]_k \leftarrow \mathsf{Decom}(o_k^{\mathsf{sk}^1}, c_k^{\mathsf{sk}^1}), \\ & \langle 0 \rangle_k \leftarrow \mathsf{Decom}(o_k^0, c_k^0), \\ & \langle y \rangle_k = (1 - \beta_k)[\mathsf{sk}^0]_k \\ & + \beta_k [\mathsf{sk}^1]_k + \langle 0 \rangle_k \end{array} \right\}.$$

Protocol Description. The formal description of our construction $\mathsf{OPTI} = \big((\mathsf{OPTI}_1^{\mathsf{R}}, \mathsf{OPTI}_2^{\mathsf{R}}), (\mathsf{OPTI}_1^{\mathsf{S}}, \mathsf{OPTI}_2^{\mathsf{S}}), ((\mathsf{OPTI}_1^k, \mathsf{OPTI}_2^k)_{k \in [n]}), \mathsf{OPTI}_{\mathsf{output}}\big)$ appears in Figure 4.

OPTI Protocol

Building Blocks:

- Shamir secret sharing ($\mathtt{share}, \mathtt{reconstruct}$), as described in Section 2.3
- A secure commitment scheme ($\mathsf{Com}, \mathsf{Decom}$), as described in Section 2.2
- ElGamal public-key encryption ($\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{KeyVer}$), as described in Section 2.1
- A multi-string NIZK scheme ($\mathsf{mNIZK.Gen}, \mathsf{Prove}, \mathsf{Verify}$), as described in Section 2.4

$\mathsf{OPTI}_1^{\mathsf{R}}$:

1. Set $b_0 = b$ and sample $b_1, \ldots, b_t, r_0, \ldots, r_t \leftarrow \mathbb{Z}_q$. Define $B(X) = \sum_{j=0}^{t} b_j X^j$ and $R_B(X) = \sum_{j=0}^{t} r_j X^j$.
2. Sample $\alpha \leftarrow \mathbb{Z}_q^*$ and set $U = g^{\alpha}$. For $j \in \{0, \ldots, t\}$, compute $(A_j, D_j) = (g^{r_j}, U^{r_j} g^{b_j})$.
3. For $k \in [n]$, compute $(\beta_k, \rho_k) = (B(k), R_B(k))$ and send (β_k, ρ_k) to $\mathcal{P}_k$. Broadcast $\Gamma_{\mathsf{R}} = (U, \{(A_j, D_j)\}_{j=0}^{t})$.

$\mathsf{OPTI}_1^{\mathsf{S}}$:

1. Sample $\mathsf{sk}^0 \leftarrow \mathbb{Z}_q$ and $\mathsf{sk}^1 \leftarrow \mathbb{Z}_q \setminus \{\mathsf{sk}^0\}$.
2. Compute $([\mathsf{sk}^0]_1, \ldots, [\mathsf{sk}^0]_n) \leftarrow \mathtt{share}(\mathsf{sk}^0, n, t+1; \rho^{\mathsf{sk}^0})$.
   Compute $([\mathsf{sk}^1]_1, \ldots, [\mathsf{sk}^1]_n) \leftarrow \mathtt{share}(\mathsf{sk}^1, n, t+1; \rho^{\mathsf{sk}^1})$.
   Compute $(\langle 0 \rangle_1, \ldots, \langle 0 \rangle_n) \leftarrow \mathtt{share}(0, n, 2t+1; \rho^0)$.
3. Compute $(o^{x_0}, c^{x_0}) \leftarrow \mathsf{Com}(x_0)$.
   Compute $(o^{x_1}, c^{x_1}) \leftarrow \mathsf{Com}(x_1)$.
4. For $k \in [n]$:
   - Compute $(o_k^{\mathsf{sk}^0}, c_k^{\mathsf{sk}^0}) \leftarrow \mathsf{Com}([\mathsf{sk}^0]_k)$.
   - Compute $(o_k^{\mathsf{sk}^1}, c_k^{\mathsf{sk}^1}) \leftarrow \mathsf{Com}([\mathsf{sk}^1]_k)$.
   - Compute $(o_k^0, c_k^0) \leftarrow \mathsf{Com}(\langle 0 \rangle_k)$.
   - Send $(([\mathsf{sk}^0]_k, o_k^{\mathsf{sk}^0}), ([\mathsf{sk}^1]_k, o_k^{\mathsf{sk}^1}), (\langle 0 \rangle_k, o_k^0))$ to $\mathcal{P}_k$.
5. Broadcast $(c^{x_0}, c^{x_1}, \{(c_k^{\mathsf{sk}^0}, c_k^{\mathsf{sk}^1}, c_k^0)\}_{k \in [n]})$.

OPTI_1^{k}:

1. Compute $(\mathsf{sk}_k, \mathsf{pk}_k) \leftarrow \mathsf{KeyGen}(\rho_k^{\mathsf{gen}})$.
2. Compute $\mathtt{crs}_k \leftarrow \mathsf{mNIZK.Gen}(\rho_k^{\mathtt{crs}})$.
3. Broadcast $(\mathsf{pk}_k, \mathtt{crs}_k)$.

$\mathsf{OPTI}_2^{\mathsf{R}}$:

1. Set $\overline{\mathtt{crs}} = (\mathtt{crs}_1, \ldots, \mathtt{crs}_n)$.
2. Set $\phi_{\mathsf{rec}} = \Gamma_{\mathsf{R}}, w_{\mathsf{rec}} = (\alpha, \{(b_j, r_j)\}_{j=0}^{t})$.
3. Broadcast $\pi_{\mathsf{rec}} \leftarrow \mathsf{Prove}(\overline{\mathtt{crs}}, \phi_{\mathsf{rec}}, w_{\mathsf{rec}})$.

$\mathsf{OPTI}_2^{\mathsf{S}}$:

1. For $k \in [n]$:
   - Compute $ct_k^{\mathsf{sk}^0} \leftarrow \mathsf{Enc}(\mathsf{pk}_k, [\mathsf{sk}^0]_k; \rho_{\mathsf{enc},k}^{\mathsf{sk}^0})$.
   - Compute $ct_k^{\mathsf{sk}^1} \leftarrow \mathsf{Enc}(\mathsf{pk}_k, [\mathsf{sk}^1]_k; \rho_{\mathsf{enc},k}^{\mathsf{sk}^1})$.
   - Compute $ct_k^0 \leftarrow \mathsf{Enc}(\mathsf{pk}_k, \langle 0 \rangle_k; \rho_{\mathsf{enc},k}^0)$.
2. Sample $\sigma \leftarrow \{0,1\}$. For $c \in \{0,1\}$:
   - Set $\mathsf{pk}_{c\oplus\sigma}^{\mathsf{slot}} = g^{\mathsf{sk}^c}$.
   - Compute $ct_{c\oplus\sigma}^{\mathsf{slot}} \leftarrow \mathsf{Enc}(\mathsf{pk}_{c\oplus\sigma}^{\mathsf{slot}}, x_c; \rho_{\mathsf{slot}}^c)$.
3. Set $\overline{\mathtt{crs}} \leftarrow (\mathtt{crs}_1, \ldots, \mathtt{crs}_n)$.
4. Set ϕ_{send} and w_{send} to the public and private values, respectively, specified in the definition of $\mathcal{R}_{\mathsf{send}}$, and compute $\pi_{\mathsf{send}} \leftarrow \mathsf{Prove}(\overline{\mathtt{crs}}, \phi_{\mathsf{send}}, w_{\mathsf{send}})$.
5. Broadcast $(\{(ct_k^{\mathsf{sk}^0}, ct_k^{\mathsf{sk}^1}, ct_k^0)\}_{k\in[n]}, \{(\mathsf{pk}_d^{\mathsf{slot}}, ct_d^{\mathsf{slot}})\}_{d\in\{0,1\}}, \pi_{\mathsf{send}})$.

OPTI_2^{k}:

1. Set $\overline{\mathtt{crs}} \leftarrow (\mathtt{crs}_1, \ldots, \mathtt{crs}_n)$.
2. Set $\mathsf{valid}_{\mathsf{S},k} = 1$ if and only if $\mathcal{P}_k$ received $(([\mathsf{sk}^0]_k, o_k^{\mathsf{sk}^0}), ([\mathsf{sk}^1]_k, o_k^{\mathsf{sk}^1}), (\langle 0 \rangle_k, o_k^0))$ from S in the first round and all three values correctly open their corresponding public commitments. Otherwise, set $\mathsf{valid}_{\mathsf{S},k} = 0$.
3. Set $\mathsf{valid}_{\mathsf{R},k} = 1$ if and only if $\mathcal{P}_k$ received $(\beta_k, \rho_k) \in \mathbb{Z}_q^2$ from R in the first round and $\mathsf{ValidEval}(\Gamma_{\mathsf{R}}, k, \beta_k, \rho_k) = 1$. Otherwise, set $\mathsf{valid}_{\mathsf{R},k} = 0$.
4. Party $\mathcal{P}_k$ for whom $\mathsf{valid}_{\mathsf{S},k} = \mathsf{valid}_{\mathsf{R},k} = 1$:
   (a) Compute and broadcast $\langle y \rangle_k = (1-\beta_k)[\mathsf{sk}^0]_k + \beta_k[\mathsf{sk}^1]_k + \langle 0 \rangle_k$.
   (b) Set $\phi_{\mathsf{comp}}^{\langle y \rangle_k} = (\langle y \rangle_k, \Gamma_{\mathsf{R}}, k, c_k^{\mathsf{sk}^0}, c_k^{\mathsf{sk}^1}, c_k^0)$,
   $w_{\mathsf{comp}}^{\langle y \rangle_k} = ((\beta_k, \rho_k), ([\mathsf{sk}^0]_k, o_k^{\mathsf{sk}^0}), ([\mathsf{sk}^1]_k, o_k^{\mathsf{sk}^1}), (\langle 0 \rangle_k, o_k^0))$.
   (c) Broadcast $\pi_{\mathsf{comp}}^{\langle y \rangle_k} \leftarrow \mathsf{Prove}(\overline{\mathtt{crs}}, \phi_{\mathsf{comp}}^{\langle y \rangle_k}, w_{\mathsf{comp}}^{\langle y \rangle_k})$.
5. Party $\mathcal{P}_k$ for whom $\mathsf{valid}_{\mathsf{R},k} = 1$ and $\mathsf{valid}_{\mathsf{S},k} = 0$:
   (a) Broadcast $((\beta_k, \rho_k), \mathsf{sk}_k, \mathsf{abort}_{\mathsf{S}})$.
6. Party $\mathcal{P}_k$ for whom $\mathsf{valid}_{\mathsf{S},k} = 1$ and $\mathsf{valid}_{\mathsf{R},k} = 0$:
   (a) Broadcast $\mathsf{abort}_{\mathsf{R}}$.
7. Party $\mathcal{P}_k$ for whom $\mathsf{valid}_{\mathsf{S},k} = \mathsf{valid}_{\mathsf{R},k} = 0$:
   (a) Broadcast $(\mathsf{abort}_{\mathsf{S}}, \mathsf{abort}_{\mathsf{R}})$.

$\mathsf{OPTI}_{\mathsf{output}}$:

1. If more than t parties sent $\mathsf{abort}_{\mathsf{S}}$ or $\mathsf{abort}_{\mathsf{R}}$, output the corresponding abort.
2. Check $\mathsf{Verify}(\overline{\mathtt{crs}}, \phi_{\mathsf{rec}}, \pi_{\mathsf{rec}}) = 1$ and output $\mathsf{abort}_{\mathsf{R}}$ otherwise.
3. Check $\mathsf{Verify}(\overline{\mathtt{crs}}, \phi_{\mathsf{send}}, \pi_{\mathsf{send}}) = 1$ and output $\mathsf{abort}_{\mathsf{S}}$ otherwise.
4. Let $\mathcal{S}_{\mathsf{S}}, \mathcal{S}_{\mathsf{R}}, \mathcal{S}_{\mathsf{S,R}}, \mathcal{S}_{\emptyset}$ denote sets of indices of parties who aborted blaming the receiver, who aborted blaming the sender, who provided output, and who aborted blaming both, respectively.
5. For $k \in \mathcal{S}_{\mathsf{S,R}}$: check $\mathsf{Verify}(\overline{\mathtt{crs}}, \phi_{\mathsf{comp}}^{\langle y \rangle_k}, \pi_{\mathsf{comp}}^{\langle y \rangle_k}) = 1$ and remove k from $\mathcal{S}_{\mathsf{S,R}}$ otherwise.
6. For $l \in \mathcal{S}_{\mathsf{R}}$:
   - Check $\mathsf{ValidEval}(\Gamma_{\mathsf{R}}, l, \beta_l, \rho_l) = 1$ and $\mathsf{KeyVer}(\mathsf{pk}_l, \mathsf{sk}_l) = 1$, and remove l from $\mathcal{S}_{\mathsf{R}}$ otherwise.
   - Compute $[\mathsf{sk}^0]_l \leftarrow \mathsf{Dec}(\mathsf{sk}_l, ct_l^{\mathsf{sk}^0})$.
   - Compute $[\mathsf{sk}^1]_l \leftarrow \mathsf{Dec}(\mathsf{sk}_l, ct_l^{\mathsf{sk}^1})$.
   - Compute $\langle 0 \rangle_l \leftarrow \mathsf{Dec}(\mathsf{sk}_l, ct_l^0)$.
   - Compute $\langle y \rangle_l = (1 - \beta_l)[\mathsf{sk}^0]_l + \beta_l[\mathsf{sk}^1]_l + \langle 0 \rangle_l$.
7. If $|\mathcal{S}_{\mathsf{S,R}}| + |\mathcal{S}_{\mathsf{R}}| < 2t + 1$, output $\mathsf{abort}_{\mathsf{R}}$.
8. Compute $y \leftarrow \mathtt{reconstruct}(\{\langle y \rangle_k\}_{k \in \mathcal{S}_{\mathsf{S,R}}}, \{\langle y \rangle_l\}_{l \in \mathcal{S}_{\mathsf{R}}})$.
9. Define $\mathcal{D}(y) = \{d \in \{0,1\} : \mathsf{pk}_d^{\mathsf{slot}} = g^y\}$. If $\mathcal{D}(y) = \emptyset$, output $\mathsf{abort}_{\mathsf{R}}$.
10. Let $\mathcal{D}(y) = \{d\}$. Compute $x \leftarrow \mathsf{Dec}(y, ct_d^{\mathsf{slot}})$. If $x = \perp$, output $\mathsf{abort}_{\mathsf{S}}$; otherwise, output x.

Fig. 4: OPTI Protocol

Theorem 1. *The OPTI construction in Figure 4 is secure provided that the DDH assumption holds (and thus the ElGamal encryption scheme* (KeyGen, Enc, Dec, KeyVer) *— described in Section 2.1) — is secure),* (Com, Decom) *is a secure commitment scheme (Section 2.2), and* (mNIZK.Gen, Prove, Verify) *is a secure multi-string NIZK scheme (Section 2.4).*

(Note that the construction also relies on Shamir secret sharing; however, we don't list it as an assumption in the theorem, as it is unconditionally secure.)

We provide the proof of our construction satisfying sender privacy, privacy and identifiability in the full version.

3.3 Communication Complexity

We now analyze the communication complexity of the OPTI construction in Figure 4. Let $\mathcal{CC}_1$, $\mathcal{CC}_2$, and $\mathcal{CC}$ denote the first-round, second-round, and total

communication complexities, respectively. In the first round, the sender communicates two degree-t key sharings, one degree-$2t$ zero-sharing, their corresponding commitments, and two input commitments. The receiver broadcasts $t+1$ coefficient pairs and privately sends one constant-size evaluation pair to each intermediary. This contributes $O(n\lambda)$ communication because $t < n$. The public keys and CRS strings broadcast by the intermediaries are subsumed by the same bound.

In the second round, the main contribution consists of the sender's recovery ciphertexts, its two public input slots and proof π_{send}, the receiver's proof of well-formedness and bitness of its coefficient transcript, and the intermediaries' local computation proofs. Thus, $\mathcal{CC}_2 = \text{poly}(n, \lambda)$ and the total communication complexity is $\mathcal{CC} = \text{poly}(n, \lambda)$.

4 Definition of MPC with Identifiable Abort

We follow the real-world/ideal-world simulation paradigm and we adopt the security model of Cohen, Garay and Zikas [CGZ20].

Real world. An n-party protocol $\Pi = (\mathcal{P}_1, \ldots, \mathcal{P}_n)$ is an n-tuple of probabilistic polynomial-time (PPT) interactive Turing machines (ITMs), where each party $\mathcal{P}_i$ is initialized with input $x_i \in \{0,1\}^*$ and random coins $\rho_i \in \{0,1\}^*$. We let $\mathcal{A}$ denote a special PPT ITM that represents the adversary and that is initialized with input that contains the identities of the corrupt parties, their respective private inputs and an auxiliary input.

The protocol is executed in rounds (i.e., the protocol is synchronous). Each round consists of a send phase and a receive phase, where parties can respectively send and receive messages. The communication in each round happens via a broadcast channel. During the execution of the protocol, the corrupt parties receive arbitrary instructions from the adversary $\mathcal{A}$, while the honest parties faithfully follow the instructions of the protocol. We consider the adversary $\mathcal{A}$ to be rushing, i.e., during every round, the adversary can see the messages the honest parties sent before producing messages for corrupt parties.

At the end of the protocol execution, the honest parties produce their outputs, and the adversary returns an arbitrary function of the corrupt parties' view. The view of a party during the execution consists of its input, random coins and the messages it sees during the execution.

Definition 10 (Real-world execution). *Let $\Pi = (\mathcal{P}_1, \ldots, \mathcal{P}_n)$ be an n-party protocol and let $\mathcal{I} \subseteq [n]$, of size at most t, denote the set of indices of the parties corrupted by $\mathcal{A}$. The joint execution of Π under $(\mathcal{A}, \mathcal{I})$ in the real world, on input vector $\mathbf{x} = (x_1, \ldots, x_n)$, auxiliary input aux and the unary representation of the security parameter 1^λ, denoted $REAL_{\Pi,\mathcal{I},\mathcal{A}(\mathsf{aux})}(\mathbf{x}, 1^\lambda)$, is defined as the output vector of $\mathcal{P}_1, \ldots, \mathcal{P}_n$ and $\mathcal{A}(\mathsf{aux})$ resulting from the protocol interaction.*

Ideal world.

Definition 11 (Ideal-world Computation). *Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party function and let $\mathcal{I} \subseteq [n]$, of size at most t, be the set of indices of the corrupt parties. Then, the joint ideal execution of f under $(\mathcal{S}, \mathcal{I})$ on input vector $\mathbf{x} = (x_1, \ldots, x_n)$, auxiliary input* aux *to $\mathcal{S}$ and the unary representation of the security parameter 1^λ, denoted $IDEAL^{\mathsf{id\text{-}abort}}_{f,\mathcal{I},\mathcal{S}(\mathsf{aux})}(\mathbf{x}, 1^\lambda)$, is defined as the output vector of $\mathcal{P}_1, \ldots, \mathcal{P}_n$ and $\mathcal{S}$ resulting from the following ideal process.*

1. Parties send inputs to trusted party*: An honest party $\mathcal{P}_i$ sends its input x_i to the trusted party. The simulator $\mathcal{S}$ may send to the trusted party arbitrary inputs for the corrupt parties. Let x'_i be the value actually sent as the input of party $\mathcal{P}_i$.*
2. Trusted party speaks to simulator*: The trusted party computes $(y_1, \ldots, y_n) = f(x'_1, \ldots, x'_n)$. If there are no corrupt parties proceed to step 3. Otherwise, the trusted party sends $\{y_i\}_{i \in \mathcal{I}}$ to $\mathcal{S}$.*
3. Simulator $\mathcal{S}$ responds to the trusted party*: If it chooses to abort, the simulator $\mathcal{S}$ can select a corrupt party $i^* \in \mathcal{I}$ who will be blamed, and send (abort, i^*) to the trusted party. If it does, we take $\mathcal{K} = [n] \setminus \mathcal{I}$.*

   Trusted party answers parties*:*

   (a) If the trusted party got abort *from the simulator $\mathcal{S}$,*
      i. It sets the abort message abortmsg $= (\perp, i^*)$.
      ii. The trusted party sends y_j to every party $\mathcal{P}_j$, $j \in [n] \setminus \mathcal{K}$. The trusted party sends abortmsg *to every party $\mathcal{P}_j$, $j \in \mathcal{K}$.*

   (b) Otherwise, it sends y to every $\mathcal{P}_j$, $j \in [n]$.
4. Outputs*: Honest parties always output the message received from the trusted party, while the corrupt parties output nothing. The simulator $\mathcal{S}$ outputs an arbitrary function of the initial inputs $\{x_i\}_{i \in \mathcal{I}}$, the messages received by the corrupt parties from the trusted party and its auxiliary input.*

Remark 1. The ideal-world computation for unanimous abort (UA) security is identical to the above, except that in Step 3, an aborting $\mathcal{S}$ does not provide a corrupt party index i^*. Consequently, the abortmsg is simply $\perp$ in this case.

Security Definitions. Now that we have described the real-world and the ideal-world experiment, we are ready to state the formal definition.

Definition 12. *Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party function. A protocol Π t-securely computes the function f with* id-abort *security if for every PPT real-world adversary $\mathcal{A}$ with auxiliary input* aux*, there exists a PPT simulator $\mathcal{S}$ such that for every $\mathcal{I} \subseteq [n]$ of size at most t, for all $\mathbf{x} \in (\{0,1\}^*)^n$, for all $\lambda \in \mathbb{N}$, it holds that*

$$REAL_{\Pi,\mathcal{I},\mathcal{A}(\mathsf{aux})}(\mathbf{x}, 1^\lambda) \approx_{\mathrm{c}} IDEAL^{\mathsf{id\text{-}abort}}_{f,\mathcal{I},\mathcal{S}(\mathsf{aux})}(\mathbf{x}, 1^\lambda)$$

5 Construction of Two-Round MPC with IA for $t < \frac{n}{3}$

Our approach to constructing our two-round IA protocol is to lift any two-round UA protocol in the plain model Π_{UA} (such as [ACGJ18]) to one that supports cheater identification. Our first step is to observe that the correctness of broadcast messages can be enforced via zero-knowledge proofs. As our protocol must operate in the plain model, we can not make use of standard NIZKs directly. Instead, we rely on multi-string NIZKs where the CRS is jointly sampled as part of the protocol: in the first round, each party publishes a CRS, and in the second round, proofs are generated with respect to the resulting joint CRS. Although proofs for the broadcast messages of both Rounds 1 and 2 can only be sent in the second round, this is still sufficient for cheater identification.

The main challenge is in detecting misbehavior in private channels in the first round (the second round can use only broadcast without loss of generality as discussed earlier). Consider a scenario where a corrupt $\mathcal{P}_j$ drops its private message to $\mathcal{P}_i$ in the first round. Whereas a UA protocol can handle this by having $\mathcal{P}_i$ simply broadcast 'abort' in the second round (leading all parties to abort unanimously) the IA setting requires more care. This is because other honest parties cannot tell whether $\mathcal{P}_i$ or $\mathcal{P}_j$ is corrupt in this scenario. We therefore require a mechanism that either identifies the cheating party in such scenarios, or enables the protocol to proceed without resolving this conflict explicitly—for instance by enabling $\mathcal{P}_i$ to compute its Round 2 message despite the missing private message.

We construct such a mechanism as follows: Each party $\mathcal{P}_i$ prepares a garbled circuit $\mathtt{GC}_i$, which garbles its next-message function for Round 2 of the protocol Π_{UA}. In more detail, the next-message function circuit takes as input $\mathcal{P}_i$'s private input and randomness, the Round 1 broadcast messages, and the Round 1 private messages received by $\mathcal{P}_i$ (denoted as $\{\mathsf{msg}^1_{j\to i}\}_{j\in[n]}$) and computes $\mathcal{P}_i$'s second round message of Π_{UA} accordingly. For all inputs except the private messages, $\mathcal{P}_i$ can directly provide the corresponding wire labels. However, this may not be possible for the wire labels associated with private-message bits (e.g. as in the scenario above).

It is at this stage that we employ OPTI (Section 3) for verifiable transfer of the wire labels corresponding to the private message. As an illustrative example, consider a situation in which the first round of the protocol requires $\mathcal{P}_j$ to send a one bit message $\mathsf{msg}_{j\to i}$ to $\mathcal{P}_i$ over their shared private channel. Let K^0 and K^1 be the labels on the corresponding wire in $\mathtt{GC}_i$. In the event that $\mathcal{P}_j$ withholds $\mathsf{msg}_{j\to i}$ from $\mathcal{P}_i$, we ensure that OPTI delivers this message instead. In particular $\mathcal{P}_i$ plays the sender and provides both labels (K^0, K^1) as its input, and $\mathcal{P}_j$ plays the receiver with $\mathsf{msg}_{j\to i}$ as its input. OPTI guarantees that either the correct label $K^{\mathsf{msg}_{j\to i}}$ is reconstructed publicly by all parties, or a cheating party is identified.

Therefore, if no cheater is identified, all first-round private-message bits are correctly incorporated into the garbled circuits. We additionally use multi-string NIZKs to prove correctness of garbling. As a result, every party can evaluate the garbled circuits of all parties and obtain the corresponding Round 2 messages of

Π_{UA}, ensuring correct completion of the protocol when no cheater is identified. This completes a high-level description of our protocol.

Before presenting the formal description of our IA protocol (Figure 5), we introduce some notation and describe a property we need from our garbling scheme. Specifically, we use a *garbling scheme with independent key generation* that allows us to pick keys independently of the circuit C. Therefore, we are able to first generate keys (i.e. labels) and later input them along with the circuit C into the `Garble` function. Such a scheme can be instantiated with Yao's garbling scheme [Yao86]. We give a formal definition in Definition 5 (Section 2.5).

Notation. Let $C_i(x_i, \rho_i, \{\mathsf{msg}_j^1\}_{j\in[n]}, \{\mathsf{msg}_{j\to i}^1\}_{j\in[n]})$ denote the Boolean circuit that takes party P_i's input x_i, randomness ρ_i, first-round broadcast messages $\mathsf{msg}_1^1, \ldots, \mathsf{msg}_n^1$, and first-round private messages $\{\mathsf{msg}_{j\to i}^1\}_{j\in[n]}$ and outputs msg_i^2. For simplicity, we assume that (x_i, ρ_i) consists of z bits, and each first-round (broadcast or private) message is ℓ bits long, so each circuit has $\mathsf{L} = z+2n\cdot\ell$ input bits. Note that C_i is public. Let g be the size of a garbled C_i (we assume for simplicity each C_i has the same size). We assume that C_i has the following order:

1. Input wires on positions $[z]$ correspond to $\mathcal{P}_i$'s private input x_i and randomness ρ_i.
2. For $j \in [n]$, input wires on positions$\{\, z + \ell \cdot (j-1) + \alpha \mid \alpha \in [\ell] \,\}$ correspond to $\mathcal{P}_j$'s first-round broadcast message msg_j^1. We denote this range as $\{z + 1, \ldots, \mathsf{M}\}$, where $\mathsf{M} = z + n \cdot \ell$.
3. For $j \in [n]$, input wires on positions $\{\, z+\ell\cdot(n+j-1)+\alpha \mid \alpha \in [\ell] \,\}$ correspond to $\mathcal{P}_j$'s first-round private message $\mathsf{msg}_{j\to i}^1$ sent to $\mathcal{P}_i$. We denote this range as $\{\mathsf{M}+1, \ldots, \mathsf{L}\}$.

For every $i, j \in [n]$ and $\beta \in [\ell]$, let $\Gamma_{i\to j}^{\beta}$ denote the public receiver coefficient transcript generated in the OPTI instance whose receiver input is $\mathsf{msg}_{i\to j}^1[\beta]$. Let

$$\omega_{i\to j}^{\beta} = \left(\eta_{i\to j}^{\beta}, \left\{\left(b_{i\to j,j}^{\beta}, r_{i\to j,j}^{\beta}\right)\right\}_{j=0}^{t}\right)$$

denote the corresponding witness for $\mathcal{R}_{\mathsf{rec}}$. In particular, $b_{i\to j,0}^{\beta}$ is the constant term of the receiver polynomial in this OPTI instance.

Relations for mNIZKs In our IA protocol, we use multi-string NIZK for the following relation.

$$\mathcal{R}_{\mathsf{IA}} = \left\{ \begin{array}{lll} & & \mathbf{K}_i \leftarrow \mathsf{KGen}(1^\lambda, \mathsf{L}, \rho_i^{\mathtt{GC}}) \\ & & \mathtt{GC}_i \leftarrow \mathtt{Garble}(1^\lambda, C_i, \mathbf{K}_i, \rho_i^{\mathtt{GC}}) \\ & \big(\mathtt{GC}_i, \mathsf{msg}_i^1, \{K_i^\alpha\}_{\alpha\in[z]}, & \mathsf{msg}_i^1, \{\mathsf{msg}_{i\to j}^1\}_{j\in[n]} \leftarrow \\ \phi = & \{\Gamma_{i\to j}^\beta\}_{\beta\in[\ell],\, j\in[n]}, & \mathtt{frst\text{-}msg}_i(x_i, \rho_i) \\ & \{(c_i^{(\alpha,0)}, c_i^{(\alpha,1)})\}_{\alpha\in\{z+1,\ldots,\mathsf{L}\}}\big) & \forall \alpha \in [z]: \quad K_i^\alpha = K_i^{\alpha,(x_i\|\rho_i)[\alpha]} \\ & \big(\rho_i^{\mathtt{GC}}, x_i, \rho_i, & \forall \alpha \in \{z+1,\ldots,\mathsf{L}\},\ c \in \{0,1\}: \\ w = & \{\omega_{i\to j}^\beta\}_{\beta\in[\ell],\, j\in[n]}, & \qquad K_i^{(\alpha,c)} \leftarrow \mathsf{Decom}(c_i^{(\alpha,c)}, o_i^{(\alpha,c)}) \\ & \{(o_i^{(\alpha,0)}, o_i^{(\alpha,1)})\}_{\alpha\in\{z+1,\ldots,\mathsf{L}\}}\big) & \forall \beta \in [\ell],\ j \in [n]: \\ & & \qquad \left(\Gamma_{i\to j}^\beta, \omega_{i\to j}^\beta\right) \in \mathcal{R}_{\mathsf{rec}} \\ & & \qquad \wedge\ b_{i\to j,0}^\beta = \mathsf{msg}_{i\to j}^1[\beta] \end{array} \right\}.$$

Remark 2. In the relation above, by a slight abuse of notation, the commitments to values actually refer to commitments to a secret sharing of those values (as used within the OPTI protocol). This is legitimate, since a set of commitments to shares is itself a secure commitment. We abuse notation in this way to simplify our relation statement.

Remark 3. For a label used as the sender input of an OPTI instance, $c_i^{(\alpha,c)}$ denotes the commitment to $K_i^{(\alpha,c)}$ generated during the first round of that instance. The same public commitment appears in ϕ_{send} for the corresponding OPTI instance and in ϕ_{IA}^i. Thus, the proof for $\mathcal{R}_{\mathsf{send}}$ binds the corresponding public slot to the committed label, while the proof for $\mathcal{R}_{\mathsf{IA}}$ binds that committed value to the label used in $\mathtt{GC}_i$.

Receiver inputs are treated differently. The receiver input to each OPTI instance is represented by the coefficient transcript $\Gamma_{i\to j}^\beta$. The relation $\mathcal{R}_{\mathsf{IA}}$ uses the corresponding witness $\omega_{i\to j}^\beta$ to prove that the constant coefficient committed by this transcript equals $\mathsf{msg}_{i\to j}^1[\beta]$.

Remark 4. Observe that in the protocol, each party commits to its private-message wire labels without permuting their order, while the corresponding OPTI slots are published in a uniformly random order. However, these commitments are never opened and are only used within the mNIZK proof, thereby keeping the position of the label that is later revealed hidden from the parties.

IA Protocol

Input: Each party $\mathcal{P}_i$ has input x_i and randomness ρ_i (which is split or expanded into randomness used for individual steps).

Building Blocks:

- Π_{OPTI} (Section 3.2)
- Π_{UA} is 2-round a honest-majority UA protocol securely computing f in the plain model (instantiated by [ACGJ18]), represented as a set of functions $\{\mathtt{frst\text{-}msg}_i, \mathtt{snd\text{-}msg}_i, \mathtt{output}\}_{i\in[n]}$.
- A secure commitment scheme $(\mathsf{Com}, \mathsf{Decom})$, as described in Section 2.2
- A multi-string NIZK scheme $(\mathsf{mNIZK.Gen}, \mathsf{Prove}, \mathsf{Verify})$, as described in Section 2.4
- A secure garbling scheme $(\mathsf{KGen}, \mathtt{Garble}, \mathsf{Eval})$ with independent key generation, as described in Section 2.5

Round 1: Each $\mathcal{P}_i$ does the following:

- Compute $\mathtt{crs}_i \leftarrow \mathsf{mNIZK.Gen}(\rho_i^{\mathtt{crs}})$
- Compute $\mathbf{K}_i \leftarrow \mathsf{KGen}(1^\lambda, \mathsf{L}, \rho_i^{\mathtt{GC}})$
- Let $\mathsf{msg}_i^1, \{\mathsf{msg}_{i\to j}^1\}_{j\in[n]} \leftarrow \mathtt{frst\text{-}msg}_i(x_i, \rho_i)$ denote the first-round broadcast and private messages of $\mathcal{P}_i$ in Π_{UA}.
- Set labels corresponding to own input and randomness: $K_i^\alpha = K_i^{\alpha,(x_i\|\rho_i)[\alpha]}$ for $\alpha \in [z]$.
- Compute commitments of labels corresponding to the broadcast message indices: For $\alpha \in \{z+1, \ldots, \mathsf{M}\}$,
  - compute $(o_i^{(\alpha,0)}, c_i^{(\alpha,0)}) \leftarrow \mathsf{Com}(K_i^{(\alpha,0)})$.
  - compute $(o_i^{(\alpha,1)}, c_i^{(\alpha,1)}) \leftarrow \mathsf{Com}(K_i^{(\alpha,1)})$.
- Broadcast $(\mathtt{crs}_i, \mathsf{msg}_i^1, \{K_i^\alpha\}_{\alpha\in[z]}, \{(c_i^{(\alpha,0)}, c_i^{(\alpha,1)})\}_{\alpha\in\{z+1,\ldots,\mathsf{M}\}})$.
- For each bit of $\mathsf{msg}_{j\to i}^1$ $(j \in [n])$, an OPTI instance is run with $\mathcal{P}_i$ as sender and $\mathcal{P}_j$ as receiver. Recall that the wire indices $\{\mathsf{M} + (j-1)\cdot\ell + 1, \ldots, \mathsf{M} + j\cdot\ell\}$ correspond to $\mathsf{msg}_{j\to i}^1$ in $\mathtt{GC}_i$. For each $j \in [n]$ and each $\beta \in [\ell]$,
  - Let $\alpha := \mathsf{M} + (j-1)\cdot\ell + \beta$.
  - Run round 1 of OPTI as S with inputs $(K_i^{(\alpha,0)}, K_i^{(\alpha,1)})$.
  - Let $\big((c_i^{(\alpha,0)}, c_i^{(\alpha,1)}), (o_i^{(\alpha,0)}, o_i^{(\alpha,1)})\big)$ denote the commitments and corresponding decommitments of the inputs $(K_i^{(\alpha,0)}, K_i^{(\alpha,1)})$, that are computed during round 1 of OPTI.
- For each bit of $\mathsf{msg}_{i\to j}^1$ $(j \in [n])$, an OPTI instance is run with $\mathcal{P}_i$ as receiver and $\mathcal{P}_j$ as sender. Recall that the wire indices $\{\mathsf{M} + (i-1)\cdot\ell + 1, \ldots, \mathsf{M} + i\cdot\ell\}$ correspond to $\mathsf{msg}_{i\to j}^1$ in $\mathtt{GC}_j$. For each $j \in [n]$ and $\beta \in [\ell]$:
  - Let $\alpha := \mathsf{M} + (i-1)\cdot\ell + \beta$.
  - Run round 1 of OPTI as R with input $\mathsf{msg}_{i\to j}^1[\beta]$.

- Let $\Gamma^{\beta}_{i\to j}$ denote the coefficient transcript broadcast by $\mathcal{P}_i$ in this execution of the first round of OPTI, and let $\omega^{\beta}_{i\to j} = \left(\eta^{\beta}_{i\to j}, \left\{\left(b^{\beta}_{i\to j,j}, r^{\beta}_{i\to j,j}\right)\right\}_{j=0}^{t}\right)$ denote the corresponding receiver witness generated in this execution.

– Run round 1 of OPTI as an intermediary for all $n^2 \cdot \ell$ executions.

Round 2: Each $\mathcal{P}_i$ does the following:

– Set $\overline{\mathtt{crs}} \leftarrow (\mathtt{crs}_1, ..., \mathtt{crs}_n)$.

– Broadcast $\mathtt{GC}_i \xleftarrow{\$} \mathtt{Garble}(C_i, \mathbf{K}_i, \rho_i^{\mathtt{GC}})$, where C_i denotes the circuit corresponding to $\mathtt{snd\text{-}msg}_i$.

– Set $\phi^i_{\mathsf{IA}} = (\mathtt{GC}_i, \mathsf{msg}^1_i, \{K^{\alpha}_i\}_{\alpha\in[z]}, \{\Gamma^{\beta}_{i\to j}\}_{\beta\in[\ell],j\in[n]}, \{(c_i^{(\alpha,0)}, c_i^{(\alpha,1)})\}_{\alpha\in\{z+1,...,\mathsf{L}\}})$ and $w^i_{\mathsf{IA}} = (\rho_i^{\mathtt{GC}}, x_i, \rho_i, \{\omega^{\beta}_{i\to j}\}_{\beta\in[\ell],j\in[n]}, \{(o_i^{(\alpha,0)}, o_i^{(\alpha,1)})\}_{\alpha\in\{z+1,...,\mathsf{L}\}})$.

– Broadcast $\pi^i_{\mathsf{IA}} \leftarrow \mathsf{Prove}(\overline{\mathtt{crs}}, \phi^i_{\mathsf{IA}}, w^i_{\mathsf{IA}})$.

– Compute the labels for $\mathtt{GC}_i$ corresponding to first-round broadcast messages: For each $j \in [n], \beta \in [\ell]$, let $\alpha := z + \ell \cdot (j-1) + \beta$, and set $K^{\alpha}_i = K_i^{(\alpha,\, \mathsf{msg}^1_j[\beta])}$.

– Broadcast $\{K^{\alpha}_i\}_{\alpha\in\{z+1,...,\mathsf{M}\}}$.

– Run round 2 of the OPTI instances initiated in the first round i.e. for $j \in [n]$:
- For $\beta \in [\ell]$, let $\alpha := \mathsf{M} + (j-1)\cdot\ell + \beta$, and run round 2 of OPTI as S with $\mathsf{R} = \mathcal{P}_j$.
- For $\beta \in [\ell]$, let $\alpha := \mathsf{M} + (i-1)\cdot\ell + \beta$, and run round 2 of OPTI as R with $\mathsf{S} = \mathcal{P}_j$.

– Run round 2 of OPTI as an intermediary for all $n^2 \cdot \ell$ executions.

Output: Each party $\mathcal{P}_i$ does the following:

– Run $\mathsf{OPTI}_{\mathsf{output}}$ for all $n^2 \cdot \ell$ executions.

– If there was an instance of OPTI with an output abort_j, output abort_j.

– For $j \in [n]$: check $\mathsf{Verify}(\overline{\mathtt{crs}}, \phi^j_{\mathsf{IA}}, \pi^j_{\mathsf{IA}}) = 1$ and output abort_j otherwise.

– For $j \in [n]$: compute $\mathsf{msg}^2_j \leftarrow \mathsf{Eval}(\mathtt{GC}_j, \{K^{\alpha}_j\}_{\alpha\in[\mathsf{L}]})$ and output abort_j if evaluation fails.

– Output $y \leftarrow \mathsf{output}(\{\mathsf{msg}^1_j\}_{j\in[n]}, \{\mathsf{msg}^1_{j\to i}\}_{j\in[n]}, \{\mathsf{msg}^2_j\}_{j\in[n]})$

Fig. 5: IA Protocol

We are now ready to state our final theorem and refer to the full version for the proof.

Theorem 2. *Let* $(\mathsf{Com}, \mathsf{Decom})$ *be a secure commitment scheme (Section 2.2);* $(\mathsf{mNIZK.Gen}, \mathsf{Prove}, \mathsf{Verify})$ *a secure multi-string NIZK scheme (Section 2.4);*

$(\mathsf{KGen}, \mathtt{Garble}, \mathsf{Eval})$ *a secure garbling scheme with independent key generation; and* Π_{OPTI} *be the* 2*-round OPTI protocol (Section 3.1) described in Figure 4. Let* f *be an efficiently computable function, and* Π_{UA} *be a* 2*-round protocol that securely computes* f *with unanimous abort in the presence of a malicious adversary that corrupts* $t < n/3$*. Then, the 2-round IA construction in Figure 5 securely computes* f *with identifiable abort in the presence of a malicious adversary that corrupts* $t < n/3$ *parties as per Definition 12.*

Acknowledgements

Jure Sternad and Sophia Yakoubov are supported in part by the Danish Independent Research Council under Grant-ID DFF-2032-00122B and DFF-2064-00016B (YOSO). Divya Ravi is funded by NWO Talent Programme Veni (VI.Veni.242.295).

We would like to thank the insightful Asiacrypt 2026 reviewers for helping us improve this paper.

References

ABT19. Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Degree 2 is complete for the round-complexity of malicious MPC. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 504–531, May 2019.

ABT21. Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect secure computation in two rounds. *SIAM J. Comput.*, 50(1):68–97, 2021.

ACGJ18. Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 395–424, August 2018.

ACGJ19. Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Two round information-theoretic MPC with malicious security. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 532–561, May 2019.

ACJ17. Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 468–499, August 2017.

AKP20. Benny Applebaum, Eliran Kachlon, and Arpita Patra. The round complexity of perfect MPC with active security and optimal resiliency. In *61st FOCS*, pages 1277–1284. IEEE Computer Society Press, November 2020.

AKP23. Benny Applebaum, Eliran Kachlon, and Arpita Patra. The round complexity of statistical MPC with optimal resiliency. pages 1527–1536. ACM Press, 2023.

AL10. Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *J. Cryptol.*, 23(2):281–343, 2010.

BHP17. Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 645–677, November 2017.

BL18. Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 500–532, April / May 2018.

CCG+20. Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 291–319, November 2020.

CDvdG88. David Chaum, Ivan Damgård, and Jeroen van de Graaf. Multiparty computations ensuring privacy of each party's input and correctness of the result. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 87–119, August 1988.

CGZ20. Ran Cohen, Juan A. Garay, and Vassilis Zikas. Broadcast-optimal two-round MPC. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 828–858, May 2020.

Cle86. Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *18th ACM STOC*, pages 364–369. ACM Press, May 1986.

CRSW22. Michele Ciampi, Divya Ravi, Luisa Siniscalchi, and Hendrik Waldner. Round-optimal multi-party computation with identifiable abort. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 335–364, May / June 2022.

DRSY23a. Ivan Damgård, Divya Ravi, Luisa Siniscalchi, and Sophia Yakoubov. Broadcast-optimal two round MPC with asynchronous peer-to-peer channels. In *LATINCRYPT*, volume 14168 of *Lecture Notes in Computer Science*, pages 87–106. Springer, 2023.

DRSY23b. Ivan Damgård, Divya Ravi, Luisa Siniscalchi, and Sophia Yakoubov. Minimizing setup in broadcast-optimal two round MPC. In *EUROCRYPT 2023, Part II*, LNCS, pages 129–158, June 2023.

ElG84. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18, August 1984.

GGHR14. Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 74–94, February 2014.

GIKR02. Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 178–193, August 2002.

GJPR21. Aarushi Goel, Abhishek Jain, Manoj Prabhakaran, and Rajeev Raghunath. On communication models and best-achievable security in two-round MPC. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 97–128, November 2021.

GLS15. S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 63–82, August 2015.

GMPP16. Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 448–476, May 2016.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

GO07. Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 323–341, August 2007.

GS18. Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 468–499, April / May 2018.

HLP11. Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 132–150, August 2011.

IKP10. Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 577–594, August 2010.

KO04. Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354, August 2004.

KRY24. Sebastian Kolby, Divya Ravi, and Sophia Yakoubov. Constant-round YOSO MPC without setup. *IACR Commun. Cryptol.*, 1(3):30, 2024.

MW16. Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763, May 2016.

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140, August 1992.

PR18. Arpita Patra and Divya Ravi. On the exact round complexity of secure three-party computation. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 425–458, August 2018.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.