Four Round Secure Computation without Setup

Zvika Brakerski*
Weizmann Institute of Science

Shai Halevi†
IBM

Antigoni Polychroniadou‡
Cornell Tech

Abstract

We construct a 4-round multi-party computation protocol for any functionality, which is secure against a malicious adversary. Our protocol relies on the sub-exponential hardness of the Learning with Errors (LWE) problem with polynomial noise ratio, and on the existence of adaptively secure commitments. Our round complexity matches a lower bound of Garg et al. (EUROCRYPT '16), and outperforms the state of the art of 6-rounds based on similar assumptions to ours, and 5-rounds relying on indistinguishability obfuscation.

Our construction takes after the multi-key FHE approach of Mukherjee-Wichs (EUROCRYPT '16) who constructed a 2-round semi-malicious protocol from LWE in the common random string (CRS) model. We show how to use a preliminary round of communication to replace the CRS, thus achieving 3-round semi-malicious security without setup. Adaptive commitments and zero-knowledge proofs are then used to compile the protocol into the fully malicious setting.

*Supported by the Israel Science Foundation (Grant No. 468/14), Alon Young Faculty Fellowship and Binational Science Foundation (Grant No. 712307).

†Supported by the Defense Advanced Research Projects Agency (DARPA) and Army Research Office(ARO) under Contract No. W911NF-15-C-0236.

‡Supported by the National Science Foundation under Grant No. 1617676, IBM under Agreement 4915013672, the Packard Foundation under Grant 2015-63124, and the Danish National Research Foundation and the National Science Foundation of China (under the grant 61361136003) for the Sino-Danish Center for the Theory of Interactive Computation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors.

Contents

1 Introduction

Secure Multi-party Computation (MPC) [Yao82, Yao86, GMW87] allows mutually suspicious parties to evaluate a function on their joint private inputs without revealing these inputs to each other. One particularly fruitful line of investigation is concerned with the *round complexity* of these protocols, more specifically how many rounds of broadcast are needed.

There are many variants of this question, depending on the exact model. In particular, the availability (or lack thereof) of a *Common Reference String* (CRS) plays an important role in the study of round complexity. It is known that at least two rounds of interaction are necessary even if such trusted CRS is available (e.g., [HLP11]), and several works in the literature construct optimal (two) round secure computation protocols both in the two-party and multi-party settings (see Section 1.1 for related works).

In the plain model without trusted setup, Katz and Ostrovsky [KO04] proved that five rounds are necessary and sufficient for secure computation in the two-party setting where both parties rceive output. In the multi-party setting, it is long known that *constant round protocols* are possible [BMR90, KOS03, Pas04, DI05, DI06, IPS08, PPV08, Wee10, Goy11, LP11a, GLOV12], however there is no known round optimal multi-party computation protocol.

In a recent work by Garg et al. [GMPP16], it is shown that when simultaneous messages are allowed, then the Katz-Ostrovsky lower bound drops to only *four* rounds (for either two or many parties). In terms of constructions, [GMPP16] show how to transform any t-round (parallel) non-malleable commitment into a $\max(4, t+1)$-round protocol for the specific coin-flipping functionality in the multi-party setting. Instantiating this protocol with the non-malleable commitments from [PPV08, COSV16], Garg et al. obtained a four round protocol for the multi-party *coin-flipping* functionality. Relying on this coin-flipping protocol they showed how to transform the two-round multi-party protocol for general functionalities of Garg et al. [GGHR14] based on indistinguishability Obfuscation (iO) [GGH+13] in the CRS model to a five-round protocol in the plain model, and the two-round LWE-based protocol of Mukherjee and Wichs [MW16] to a six-round protocol.

A lot of work has been done on the round complexity of non-malleable commitments as well. However, the only candidates for parallel non-malleable commitments that run in less than four rounds are constructed in the works of [PPV08, COSV16]. The two round parallel non-malleable commitment of [PPV08] is based on adaptive pseudorandom generators (PRGs) and the three round non-malleable commitment of [COSV16] is based on one-way permutations with sub-exponential security.

Even under very strong assumptions, prior works leave the following fundamental question open:

> *Can we obtain round-optimal multi-party computation protocols in the plain model (without setup)?*

We answer the above question in the affirmative, obtaining a round-optimal multi-party computation protocol in the plain model for general functionalities in the presence of a malicious adversary. Our starting point is the observation that we can replace the four-round coin-tossing from [GMPP16] by an extremely simple *single round protocol*. That protocol yeilds a common somewhat-random string, which is much weaker than a truly random string but still "good enough for LWE". This lets us run (a variant of) the Mukherjee-Wichs protocol [MW16], yeilding a three-round protocol with "semi-malicious" security, under the LWE assumption.

To get security in the malicious adversary model, we rely on other strong assumptions. In particular, we use the two-round adaptively secure commitment scheme of Pandey, Pass and Vaikuntanathan [PPV08] (which is just Naor's protocol from [Nao91], when instantiated with adaptive PRGs). Moreover, we need a sub-exponential version of the LWE assumption. Hence, we prove the following:

Theorem 1. (Informal) Assuming the existence of adaptive commitments, as well as the sub-exponential hardness of Learning-with-Errors, there exists a four-round protocol that securely realizes any multi-party functionality against a malicious adversary in the plain model without setup.

1.1 Related Work

Related work in the CRS model. The works of Jarecki and Shmatikov [JS07] and Horvitz and Katz [HK07] present *two-party* protocols where the former is constant-round and the latter is optimal (two)-round. Asharov et al. [AJL+12] first show a three-round *multi-party* computation protocol in the CRS model and a two-round multi-party computation protocol in the reusable PKI model, under LWE, by constructing threshold FHE schemes (based on the FHE schemes from [BV11, BGV12]). The works of [BD10, MSS11] also present threshold FHE schemes but their protocols required more than two rounds of interaction. The work of Garg et al. [GGHR14] gives a two-round multi-party protocol under strong assumptions, namely, the existence of indistinguishability obfuscation for polynomial circuits and statistically-sound NIZKs.

More recently, the work of [MW16], and its extensions [BP16, PS16], based on multi-key FHE [LTV12, CM15], shows how to obtain optimal 2-round constructions based on LWE and NIZKs in the CRS model.

Related work in the plain model. For the computational setting and the special case of two party computation, the semi-honest secure protocol of Yao [Yao82, Yao86, LP11b] consists of only three rounds (see Section 3). An alternative approach using randomized polynomials was also given by [IK00, AIK05]. For malicious security, the first constant round protocol based on GMW was presented by Lindell [Lin01]. The work of [IPS08] presented a different approach which also results in a constant round protocol.

The exact round complexity of two party computation was studied in the work of Katz and Ostrovsky [KO04] who provided a 5 round protocol for computing any two-party functionality. They also ruled out the possibility of a four round protocol for coin-flipping, thus completely resolving the case of two party. Recently [ORS15] constructed a 5-round protocol for the general two-party computation by only relying on *black-box* usage of the underlying assumptions.

For the multi-party setting, the exact round complexity has remained open for a long time. The work of [BMR90] gave the first constant-round non black-box protocol for honest majority (improved by the black-box protocols of [DI05, DI06]). Katz, Ostrovsky, and Smith [KOS03] constructed logarithmic-round protocols for any multi-party functionality for the dishonest majority case based on polynomial-time assumptions and constant round protocols based on exponential-time assumptions. Pass [Pas04] constructed a constant-round protocol based on polynomial-time assumption. The constant-round protocols of [KOS03, Pas04] relied on non-black-box use of the adversary's algorithm [Bar01]. Constant-round protocols making black-box use of the adversary were constructed by [PPV08, LP11a, Goy11], and making black-box use of one-way functions by Wee in $\omega(1)$ rounds [Wee10] and by Goyal in constant rounds [Goy11]. Furthermore, based on the non-malleable commitment scheme of [Goy11], the work of [GLOV12] constructs a constant-round multi-party coin-tossing protocol.

The recent work of [GMPP16] examined the exact round complexity of secure computation in the multi-party setting and proved a lower bound of four rounds for general functionalities. They also constructed six-round protocols based on LWE and adaptive PRGs and five-round protocols based on iO and adaptive PRGs.

1.2 Overview of our Protocol

The starting point for the current work is the observation that the Mukherjee-Wichs protocol in [MW16] does not seem to need the full power of the common-random string. The Mukherjee-Wichs protocol uses a multi-key homomorphic encryption based on GSW encryption [GSW13]. In that scheme, all parties must share the same matrix A before they can generate their encryption keys, and semantic security relies on A being random. The six-round protocol of Garg et al. [GMPP16] therefore spends four rounds on a coin-tossing protocol to generate the matrix A, and then two more rounds running the Mukherjee-Wichs protocol.

We begin by asking whether we can get semantic security of the encryption even when A is not completely random. For example, imagine coosing the bits of A using a defective coin-tossing protocol with a small constant bias. It is plausible that (a) such protocol can take less than four rounds, and (b) such a biased matrix A is still good enough to serve as the public matrix for LWE. This may give us a less-than-six-round protocol based on LWE. Our eventual protocol does not exactly follow this route, but the "philosophy" underlying it is the same.

To get a four-round protocol that build on the two-round Mukherjee-Wichs protocol, we must choose the public matrix A within the first two rounds. And although two rounds are not enough for coin tossing, they may still give us a "sufficiently random" public matrix A that we can use for LWE. In fact, we show that we can use a *one-round protocol* to get such a matrix A, which is good enough for (a variant of) the two-round Mukherjee-Wichs protocol.

A variant of the Mukherjee-Wichs protocol. The Mukherjee-Wichs protocol uses the multi-key FHE scheme of Clear and McGoldrick [CM15, MW16], which is based on GSW encryption [GSW13]: The scheme uses a public random matrix $A \in \mathbb{Z}^{(n-1)\times m}$ (with $m \gg n$), each party i chooses a random vector $s_i \in \mathbb{Z}_q^{n-1}$ and a short vector $e_i \in \mathbb{Z}_q^m$, then computing $b_i = e_i - s_i A$ and setting its public key as $B_i^T = (A^T | b_i^T)$ and its secret key as $t_i = (s_i, 1)$ (all modulo some modulus q).

In our protocol, we simply let each party choose some of the entries in the matrix A. We then want to argue that the resulting encryption scheme remains secure under LWE, even though some of the entries of A are chosen by the adversary. As we explain in Section 2.2, however, this is not quite true for the Mukherjee-Wichs protocol as is. Roughly, the reason is that the vector $e_i - s_i A$ may leak too much information about s_i if A is (partially) adversarial.

Fortunately, this turns out to be easy to fix: All we need to do is "flip the dimensions", making the secret key and m-vector and the public key an m-by-n matrix. This way, the secret key s_i is much longer, and we can appeal to the leakage-resilience of LWE-based encryption to argue that the underlying encryption scheme remains secure even given the leakage $s_i A$. Making this flipped scheme works requires tweaking the ciphertext dimensions and the magnitude of noise, as we explain in Section 2.1.1. But the algebraic expressions from [CM15, MW16] (and hence the properties of the multi-key FHE scheme) remain essentially unchanged.

A skeleton protocol. We use our one-round protocol for choosing A to replace the expensive preamble coin-flipping protocol of Garg et al., thus obtaining a skeleton protocol *with only three rounds*, as follows:

Round 1: CRS. Every player P_i broadcasts a single message α_i, and the collection of α_i's defines the public matrix A needed for (our variant of) the Clear-McGoldrick multi-key FHE scheme [CM15, MW16].

Round 2: Encryption. Each party generates a public/secret key-pair for the multi-key FHE, encrypts its input under these keys, and broadcasts the public key and ciphertext.

Round 3: Decryption. Each party separately evaluates the function on the encrypted inputs, then use its secret key to compute a decryption share of the resulting evaluated

ciphertext and finally broadcasts that share to everyone.

Epilogue: Output. Once all the decryption shares are received, each party can combine them to get the function value, which is the output of the protocol.

This skeleton protocol can be shows to be secure in the semi-malicious adversary model, but it is clearly *insecure* in the presence of a malicious adversary. Although the protocol can tolerate adversarial choice of the first-round messages α_i, the adversary can still violate privacy by sending invalid ciphertexts in Round 2 and observing the partial decryption that the honest players send in the next round. It can also violate correctness by sending the wrong decryption shares in the last round.

These two attacks can be countered by having the parties prove that they behaved correctly, namely that the public keys and ciphertexts in Round 2 were generated honestly, and that the decryption shares in Round 3 were obtained by faithful decryption. To be effective we need the proof of honest encryption to complete before the parties send their decryption shares (and of course the proof of honest decryption must be completed before the output phase can be produced). Hence, if we have a k-round proof of honest encryption (and a $(k+1)$-round proof of honest decryption) then we get a $(k+1)$-round protocol overall. Much of the technical difficulties in the current work are related to using 3-round proofs of honest encryptions, resulting in a 4-round protocol.

1.3 A Tale of Malleability and Extraction

To get a provable protocol, we must exhibit a simulator that can somehow extract the inputs of the adversary, so that it can send these inputs to the trusted party in order to get the function output. To that end we make the three-round proof of honest encryption a *Proof of Knowledge* (POK), and let the simulator use the knowledge extractor to get these adversarial inputs.

At the same time, we must ensure that this proof of knowledge is non-malleable, so that the extracted adversarial inputs do not change between the real protocol (in which the honest parties prove knowledge of their true input) and the simulated protocol (in which the simulator generates proofs for the honest players without knowing their true inputs). A few subtle technicalities are discussed below.

Two-round commitment with straight-line extraction. The main technical tool that we use in our proofs is the two-round aCom commitment of Pandey et al. [PPV08] (i.e., Naor's scheme [Nao91] with adaptive PRGs), that the parties use to commit to their inputs and randomness. Commitments in this scheme are marked by *tags*, and the scheme has the remarkable property of *adaptive security*: Namely, commitments with one tag are secure *even in the presence of an oracle that breaks commitments for all other tags.* Some hybrid games in our proof of security are therefore staged in a mental-experiment game where such a breaking oracle exists, providing us with straight-line (rewinding-free)

extraction of the values that the adversary commits to, while keeping the honest-party commitments secret.

However, we also need our other primitives (MFHE, POK, etc.) to remain secure in the presence of a breaking oracle, and we use complexity leveraging for that purpose: We assume that these primitives are sub-exponentially secure, and set their parameters much larger that those of the commitment scheme. This way, all these primitives remain secure even against sub-exponential time adversaries that can break the commitment by brute force. When arguing the indistinguishability of two hybrids, we reduce to the sub-exponential security of these primitives and use brute force to implement the breaking oracle in those hybrids.[1]

Delayed-input proofs. In the three-round proofs for honest encryption and in the four-round proofs for honest decryption, the statement to be proved is not defined until just before the last round of the protocols. We therefore need to use delayed-input proofs that can handle this case.

Fake proofs via Feige-Shamir. The simulator needs to fake the four-round proof of honest decryption on behalf of the honest parties, as it derives their decryption shares from the function output that it gets from the trusted party. For this purpose we use a Feige-Shamir-type four-round proof [FS90], which has a trapdoor that we extract and let us fake these proofs.

WI-POK with a trapdoor. Some steps in our proof have hybrid games in which the commitment contains the honest parties' true inputs while the encryption contains zeros. In such hybrids, the statement that the values committed to are consistent with the encryption is not true, so we need to fake that three-round proof as well.

For that purpose we use another Feige-Shamir-type trapdoor: Each party chooses a random string R, encloses $\hat{R} = f(R)$ with its first-flow message, encloses R inside the commitment aCom (together with its input and randomness) and adds the statement $\hat{R} = f(R)$ to the list of things that it proves in the three-round POK protocol.

In addition, the parties execute a second commitment protocol bCom (which is normally used to commit to zero in the real protocol), and we modify the POK statement to say that either the original statement is true, or the value committed in that second commitment bCom is a pre-image of the $\hat{R}$ value sent by the *verifier* in the first round. Letting the POK protocol be witness-indistinguishable (WI-POK), we then extract the R value from the adversary (in some hybrids), let the challenger commit to that value in the second commitment aCom, and use it as a trapdoor to fake the proof in the POK protocol.

[1] Technically we "only" need to assume standard security in a world with such a breaking oracle, which is a weaker assumption than full sub-exponential security.

We note that the second commitment bCom need not be non-malleable or adaptive, but it does need to remain secure in the presence of a breaking oracle for the first commitment. Since we already assume a 2-round adaptive commitment aCom, then we use the same scheme also for this second commitment, and appeal to its adaptive security to argue that the second commitment remains secure in the presence of a breaking oracle for the first commitment.

Public-coin proofs. In the multi-party setting, the adversary may choose to fail the proofs with some honest parties and succeed with others. We thus need to specify what honest parties do in case one of the proofs fail. The easiest solution is to use public-coin proofs with perfect completeness, and have the parties broadcast their proofs and verify them all (not only the ones where they chose the challenge). This way we ensure that if one honest party fails the proof, then all of them do.

We comment that this is not quite needed, and we could get security even if some honest parties abort while others do not. However, the argument is a little more subtle, and we forgo this direction in the current preliminary report.

1.4 The Resulting Protocol

As mentioned above, our protocol start from using a multi-key homomorphic encryption scheme and a one-round protocol for its setup, it also uses two-round adaptive commitments $\mathsf{aCom} = (\mathsf{acom}_1, \mathsf{acom}_2)$ and $\mathsf{bCom} = (\mathsf{bcom}_1, \mathsf{bcom}_2)$, a three-round WIPOK protocol $\Pi_{\text{WIPOK}} = (\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3)$ and a four-round ZK argument of knowledge $\Pi_{\text{FS}} = (\mathsf{fs}_1, \mathsf{fs}_2, \mathsf{fs}_3, \mathsf{fs}_4)$. In the following, we provide a sketch of our protocol.

Round 1: CRS, commitment & proof. Every party i broadcasts its message α_i for the one-round protocol for choosing the public matrix A. It also broadcasts the first message acom_1 of the adaptive commitment for its randomness and input, the first message p_1 of a public-coin WI-POK for a proof of the committed values (including honest encryption), and the first message fs_1 of a public-coin proof of honest decryption.

Round 2: another commitment. Each party broadcasts messages $(\mathsf{acom}_2, \mathsf{p}_2, \mathsf{fs}_2)$. In addition it broadcasts the first commitment message bcom_1 (which will be used to commit to zero).

Round 3: Encryption & proofs. The parties collects all the first round messages α_i and use them to compute the common matrix A. Then each party runs the key-generation and encryption procedures of the multi-key FHE, and broadcasts its public key and encrypted input. In the same round, each party also broadcasts messages $(\mathsf{bcom}_2, \mathsf{p}_3, \mathsf{fs}_3)$.

Round 4: Verification & decryption. Each party runs the verifier algorithm for the Π_{WIPOK} proof of honest encryption, verifying all the instances (not just those where it played the verifier in previous rounds). If all of them passed then it evaluates the function on the encrypted inputs, then use its secret key to compute a decryption share of the resulting evaluated ciphertext, and broadcasts that share to everyone. It also broadcasts the message fs_4 of the proof of honest decryption.

Epilogue: Verification & Output. Once all the decryption shares and proofs are received, each party runs the verifier algorithm for the Π_{FS} proof of honest decryption, again verifying all the instances. If all of them passed then it combines all the decryption shares to get the function value, which is the output of the protocol.

If any of the messages is missing or mal-formed, or if any of the verification algorithms fail, then the parties are instructed to immediately abort with no output.

2 Multi-Key FHE from "Dual" GSW

Notation. We denote the security parameter by κ. A function $\mu : \mathbb{N} \to \mathbb{N}$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large κ's it holds that $\mu(\kappa) < \frac{1}{p(\kappa)}$. We use the abbreviation PPT to denote probabilistic polynomial-time. We often use $[n]$ to denote the set $\{1, ..., n\}$. $d \leftarrow \mathcal{D}$ denotes the process sampling d from the distribution $\mathcal{D}$ or, if $\mathcal{D}$ is a set, a uniform choice from it. If $\mathcal{D}_1$ and $\mathcal{D}_2$ are two distributions, then $\mathcal{D}_1 \approx_{\text{s}} \mathcal{D}_2$ denotes that they are statistically close, $\mathcal{D}_1 \approx_{\text{c}} \mathcal{D}_2$ denots computationally indistinguishability, and $\mathcal{D}_1 \equiv \mathcal{D}_2$ denotes identical distributions. For a protocol Π between two parties P_i and P_j denote by $(\mathsf{p_1}^{i,j}, \ldots, \mathsf{p_t}^{i,j})$ the view of the messages in all t rounds where the subscripts (i, j) denote that the *first* message of the protocol is sent by P_i to P_j. Likewise, subscripts (j, i) denote that the *first* message of the protocol is sent by P_j to P_i.

2.1 Multi-Key FHE

An encryption scheme is multi-key homomorphic if it can evaluate circuits on ciphertexts encrypted under different public keys. To decrypt an evaluated ciphertext, the algorithm uses the secret keys of all parties whose ciphertexts took part in the computation. In more detail, a multi-key homomorphic encryption scheme consists of five procedures, $\mathsf{MFHE} = (\mathsf{FHE.Setup}, \mathsf{MFHE.Keygen}, \mathsf{MFHE.Encrypt}, \mathsf{MFHE.Decrypt}, \text{MFHE.Eval})$:

- **Setup** $\mathsf{params} \leftarrow \mathsf{FHE.Setup}(1^\kappa)$**:** On input the security parameter κ the setup algorithm outputs the system parameters params.
- **Key Generation** $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{MFHE.Keygen}(\mathsf{params})$**:** On input params the key generation algorithm outputs a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$.

- **Encryption** $c \leftarrow \mathsf{MFHE.Encrypt}(pk, \mu)$**:** On input pk and a plaintext message $x \in \{0,1\}^*$ output a "fresh ciphertext" c. (We assume for convenience that the ciphertext includes in it also the respective public key.)
- **Evaluation** $\hat{c} := \mathrm{MFHE.Eval}(\mathsf{params}; \mathcal{C}; (c_1, \ldots, c_\ell))$**:** On input a (description of a) Boolean circuit $\mathcal{C}$ and a sequence of ℓ fresh ciphertexts $(c_1, \ldots, c_\ell)$, output an "evaluated ciphertext" $\hat{c}$. (Here we assume that the evaluated ciphertext includes also all the public keys from the c_i's.)
- **Decryption** $x := \mathsf{MFHE.Decrypt}((\mathsf{sk}_1, \ldots, \mathsf{sk}_N), \hat{c})$**:** On input an evaluated ciphertext c (with N public keys) and a the corresponding N secret keys $(\mathsf{sk}_1, \ldots, \mathsf{sk}_N)$, output the message $x \in \{0,1\}^*$.

The scheme is correct if for every circuit $\mathcal{C}$ on N inputs and any input sequence $x_1, \ldots, x_N$ for $\mathcal{C}$, if we set $\mathsf{params} \leftarrow \mathsf{FHE.Setup}(1^\kappa)$ and then generate N key-pairs and N ciphertexts $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{MFHE.Keygen}(\mathsf{params})$ and $c_i \leftarrow \mathsf{MFHE.Encrypt}(pk_i, x_i)$, then we get

$$\mathsf{MFHE.Decrypt}\big((\mathsf{sk}_1, \ldots, \mathsf{sk}_N), \mathrm{MFHE.Eval}(\mathsf{params}; \mathcal{C}; (c_1, \ldots, c_N))\big) = \mathcal{C}(x_1, \ldots, x_N)]$$

except with negligible probability (in κ) taken over the randomness of all these algorithms. We typically consider a slightly weaker notion of homomorphism, where the **Setup** algorithm gets also a depth-bound d and correctness is then defined only relative to circuits of depth upto d.

Local decryption. A special property of the multi-key FHE schemes from [CM15, MW16] that we need, is that the decryption procedure consists of a "local" partial-decryption procedure $ev_i \leftarrow \mathsf{MFHE.PartDec}(\hat{c}, \mathsf{sk}_i)$ that only takes one of the secret keys and outputs a partial decryption share, and a public combination procedure $\mu \leftarrow \mathsf{MFHE.FinDec}(ev_1, \ldots, ev_N, \hat{c})$ that takes these partial shares and outputs the plaintext.

Simulated decryption shares. Another property of the schemes from [CM15, MW16] that we need is the ability to simulate the decryption shares. Specifically, there exists a PPT simulator $\mathcal{S}^T$, that gets for input:
– the evaluated ciphertext $\hat{c}$,
– the output plaintext $x := \mathsf{MFHE.Decrypt}((\mathsf{sk}_1, \ldots, \mathsf{sk}_N), \hat{c})$,
– a subset $I \subset [N]$, and all secret keys *except the one for* I, $\{\mathsf{sk}_j\}_{j \in [N] \setminus I}$.
The simulator produces as output simulated partial evaluation decryption shares: $\{\widetilde{ev}_i\}_{i \in I} \leftarrow \mathcal{S}^T(x, \hat{c}, I, \{\mathsf{sk}_j\}_{j \in [N] \setminus I})$. We need the simulated share be indistinguishable from the shares produced by the local partial decryption procedures using the keys $\{sk_i\}_{i \in I}$ (even to a distinguisher that sees all the inputs of $\mathcal{S}^T$). We say that a scheme is *simulatable* if it has local decryption and a simulator as described here.

Security of multikey FHE. Security is defined as the usual notion of semantic security, but for our purposes we need a special variant of this notion, since we will be using a partially adversarial setup, see Section 2.1.2.

2.1.1 A "Dual" LWE-Based Multi-Key FHE

For our protocol we use a "dual" of the Clear-McGoldrick multi-key FHE scheme from [CM15, MW16]. Just like the "primal" version, our scheme uses the GSW FHE scheme [GSW13], and its security is based on the hardness of LWE.

Recall that the LWE problem is parametrized by integers n, m, q (with $m > n \log q$) and a a distributions χ over $\mathbb{Z}$ that produces whp integers much smaller than q. The LWE assumption says that given a random matrix $A \in \mathbb{Z}_q^{n \times m}$, the distribution $sA + e$ with random $s \in \mathbb{Z}_q^n$ and $e \leftarrow \chi^m$ is indistinguishable from uniform in $\mathbb{Z}_q^m$.

For the "dual" GSW scheme below, we use parameters $n < m < w < q$ with $m > n \log q$ and $w > m \log q$, and two error distriubtions χ, χ' with χ' producing much larger errors than χ (but still much smaller than q). Specifically, consider the distribution

$$\chi'' = \{a, b \leftarrow \chi^m, c \leftarrow \chi', \text{ output } \langle a, b \rangle + c\}.$$

We need the condition that the statistical distance between χ' and χ'' is negligible (in the security parameter n). This condition holds, for example, if χ, χ' as discrete Gaussian distributions around zero with parameters p, p', respectively, such that p'/p^2 is super-polynomial (in n).

Key generation. In the "dual" GSW scheme that we use, the public key of party i is matrix $B_i \in \mathbb{Z}_q^{m \times n}$, and the corresponding secret key is a *random low-norm vector* $t_i \in \mathbb{Z}_q^m$, such that $t_i B_i = 0 \pmod q$. In more detail, for $A \in \mathbb{Z}^{(m-1) \times n}$ and a low-norm vector $s_i \leftarrow \chi^{m-1}$, we set $b_i = s_i A \bmod q$ and $B_i = \begin{pmatrix} A \\ -b_i \end{pmatrix}$. Denoting $t_i = (s_i, 1)$, we indeed have $t_i B_i = 0 \pmod q$.

Just like in the multi-key scheme from [CM15, MW16], to get a multi-key FHE scheme from the "dual" GSW acheme above, we will need all the parties to share the same $(m-1)$-by-n matrix A for key-generation.

Encryption. To encrypt a bit μ under the public key B_i, choose a random matrix $R \in \mathbb{Z}_q^{n \times w}$ and a low-norm error matrix $E \in \mathbb{Z}_q^{m \times w}$, and set $C = B_i R + E + \mu G$ where G is a fixed m-by-w "gadget matrix" (whose structure is not important for us here).

For our protocol, we use more error for the last row of the error matrix E than for the top $m-1$ rows. Namely, we choose $\hat{E} \leftarrow \chi^{(m-1) \times w}$ and $e' \leftarrow \chi'^w$ and set $E = \begin{pmatrix} \hat{E} \\ e' \end{pmatrix}$.

Decryption. Just like in the GSW scheme, the invariant satisfied by ciphertexts in this scheme is that an encryption of a bit μ relative to secret key t_i is a matrix C that satisfies $tC = \mu \cdot tG + e \pmod q$ for a low-norm error matrix E, where G is a fixed m-by-w "gudget matrix" (cf. [MP12]). This invariant holds for freshly encrypted ciphertexts since $tB_i = 0 \pmod q$, and so

$$t_i(B_i R + E + \mu G) = \mu \cdot t_i G + t_i E \pmod q,$$

where $e = t_i E$ has low norm (as both t_i and E have low norm).

To decrypt, the secret-key holder computes $u = t_i \cdot C \bmod q$, outputting 1 if the result is closer to $t_i G$ or 0 is the result is closer to 0.

Since ciphertexts satisfy the same invariant as in the original GSW scheme, then the homomorphic operations in GSW work just as well for this "dual" variant. Similarly the ciphertext-extension technique from [CM15, MW16] works also for this variant exactly as it does for the "primal" scheme (see below). Hence we get a multi-key FHE scheme.

The ciphertext-expansion procedure. For the ciphertext-extension technique, recall that the there exists a low-norm vector u such that $Gu = (0, 0, \ldots, 0, 1)$, and therefore for every secret key $t = (s|1)$ we have $tGu = 1 \pmod q$. It follows that if C is an encryption of μ wrt secret key $t = (s|1)$, then the vector $v = Cu$ satisfies

$$\langle t, v\rangle = tCu = (\mu tG + e)u = \mu tGu + \langle e, u\rangle = \mu + \epsilon \pmod q$$

where ϵ is a small integer. In other words, given an encryption of μ wrt t we can construct a vector v such that $\langle t, v\rangle \approx \mu \pmod q$.

Next, let $t_1 = (s_1|1)$, $t_2 = (s_2|1)$ be two secret keys with corresponding public keys $B_1 = \begin{pmatrix} A \\ -s_1 A \end{pmatrix}$ and $B_2 = \begin{pmatrix} A \\ -s_2 A \end{pmatrix}$, then given these two public keys we can compute the vector $\delta = (s_1 - s_2)A \bmod q$.

Let $C = B_1 R + E + \mu G$ be fresh encryption of μ wrt t_1, and suppose that we also have en encryption under t_1 of the matrix R. Knowing the vector δ, we can apply homomorphic operations to the encryptiion of R to get an encryption of the entries of the vector $\rho = \delta R$, and then using the technique above we can compute for every entry ρ_i a vector x_i such that $\langle t_1, x_i\rangle \approx \rho_i \pmod q$. Concatenating all these vectors we get a matrix X such that $t_1 X \approx \rho = (s_1 - s_2)AR \pmod q$.

Finally, consider the matrix $C' = \begin{pmatrix} C & X \\ 0 & C \end{pmatrix}$, we claim that this is an encryption of the same plaintext μ under the concatenated secret key $t' = (t_1|t_2)$. To see this, notice that

$$t_2 C = (s_2|1)\left(\begin{pmatrix} A \\ -s_1 A \end{pmatrix} R + E + \mu G\right) \approx (s_2 - s_1)AR + \mu t_2 G \pmod q,$$

and therefore

$$\begin{aligned} t'C' &= (t_1 C \mid t_1 X + t_2 C) \approx (\mu t_1 G \mid (s_1 - s_2)AR + (s_2 - s_1)AR + \mu t_2 G) \\ &= \mu(t_1 G \mid t_2 G) = \mu(t_1|t_2)\begin{pmatrix} G & \\ & G \end{pmatrix}, \end{aligned}$$

as needed. As in the schemes from [CM15, MW16], this technique can be generalized to extend the ciphertext C into an encryption of the same plaintext μ under the concatenation of any number of keys.

A simulatable scheme. Exactly as for the "primal" case, the decryption shares in this "dual" scheme are only simulatable when all parties but one are corrupted. But the (generic) solution of Mukherjee-Wichs [MW16, Thm. 6.5] can be used to make this scheme simulatable for any adversary structure.

2.1.2 Our Interactive Setup Procedure and its Security

Below we prove that this "dual" GSW scheme provides semantic security not only when the matrix A is random, but even when it is partially adevrsarial. In our setting, the public matrix A will be constructed in the first round of the protocol. Namely, every party broadcasts one message, and then everyone locally computes the (same) matrix A from all these broadcast messages. Below we formulate the security property that we need in terms of a generic one-round protocol Π_{GenSetup} for choosing A, and then describe a specific protocol that realizes this security property under LWE.

Interactive LWE. Fix the number of parties N, LWE parameters n, m, w, q and error distributions χ, χ'. Let $\Pi_{\mathsf{GenSetup}} = (\mathsf{Gen}_1, \mathsf{Gen}_2)$ be an N-party, one-broadcast-round protocol with no inputs. Namely, first each party i computes a message $\alpha_i \leftarrow \mathsf{Gen}_1(n, m, w, q, N)$ (where Gen_1 is a randomized procedure), and broadcasts it to everyone. Upon receipt of all the boradcast messages α_i, each party computes a matrix $A \leftarrow \mathsf{Gen}_2(\alpha_1, \ldots, \alpha_N) \in \mathbb{Z}_q^{(n-1)\times m}$ (where Gen_2 is deterministic). Since all the parties see the same α_i's and Gen_2 is deterministic, then they all output the same matrix A.

Consider now the experiment of executing the protocol Π_{GenSetup} in the presence of a rushing adversary rA that controls all the parties but one, and then using the resulting A in the dual GSW scheme. Specifically, consider the following experiment:

Experiment $\mathsf{ILWE}(\mathsf{rA})$:

- The adversary chooses one party $i \in [N]$;
- Party P_i computes $\alpha_i \leftarrow \mathsf{Gen}_1(n, m, w, q, N)$;
- The adversary sets all the other messages $\{\alpha_j\}_{j\neq i} \leftarrow \mathsf{rA}(\alpha_i)$;
- P_i computes $A \leftarrow \mathsf{Gen}_2(\alpha_1, \ldots, \alpha_N) \in \mathbb{Z}_q^{(m-1)\times n}$, then chooses a short secret key $s_i \leftarrow \chi^{m-1}$ and sets $b_i = s_i A \bmod q \in \mathbb{Z}_q^n$.

  P_i also chooses a random bit $\sigma \in \{0, 1\}$ and proceeds as follows:

  - If $\sigma = 0$ it chooses a uniform random matrix $U \in \mathbb{Z}_q^{(m-1)\times w}$ and a uniform vector $v \in \mathbb{Z}_q^w$
  - If $\sigma = 1$ then P_i chooses also encryption randomness $R \in \mathbb{Z}_q^{n\times w}$, $\hat{E} \leftarrow \chi^{(m-1)\times w}$ and $e' \leftarrow \chi'^{w}$, and sets $U = A \times R + \hat{E} \bmod q$ and $v = b_i \times R + e' \bmod q$.

In either case, P_i sends to the adversary the tripe (b_i, U, v).

- The adversary outputs a guess σ' for the value of the bit σ.

We write $(\sigma, \sigma') \leftarrow \mathsf{ILWE}(\mathsf{r}\mathcal{A})$ to denote a run of this experiment where P_i chooses σ and the adversary outputs σ'.

Definition 1 (Interactive-LWE)**.** A protocol Π is said to be ILWE-hard (relative to the parameters n, m, w, q, χ, χ' and N) if for any PPT adversary $\mathsf{r}\mathcal{A}$ we have

$$\Pr[\sigma = \sigma' : (\sigma, \sigma') \leftarrow \mathsf{ILWE}(\mathsf{r}\mathcal{A})] \leq 1/2 + \text{negligible}(n).$$

Π is sub-exponential ILWE-hard if the same holds even for adversaries running in time 2^{n^ϵ}, for some constant $\epsilon > 0$.

2.1.3 An ILWE-Hard Protocol Under LWE

Our ILWE-hard protocol is very simple: Setting $n = N \cdot n'$ for security parameter n', we let each party i choose its own $(m-1) \times n'$ matrix A_i, then concatenate all these matrices to get $A = (A_1 | A_2 | \ldots | A_N) \in \mathbb{Z}_q^{(m-1) \times n}$. Denote this simple protocol Π_{concat}, then we have:

Theorem 1. *The protocol Π_{concat} is ILWE hard under the LWE hardness assumption with parameters n', w, q and error χ. Similarly, it is sub-exponential ILWE-hard under the sub-exponential LWE hardness assumption.*

Proof: (sketch) Proving indistinguishability of the cases $\sigma = 0, \sigma = 1$, goes through a few hybrid games:

Game H_0. The first game is the case $\sigma = 1$, where we set $b_i = s_i A$, $U = AR + \hat{E}$, and $v = b_i R + e'$ (all modulo q).

Game H_1. In the second game we compute $v = s_i U + e'$ instead of $v = b_i R + e'$.

By our choise of error distributions χ, χ', we have that $s_i U + e' = s_i(AR + \hat{E}) + e'$ is statistically close to $b_i R + e' = s_i AR + e'$ (since $s_i \hat{E} + e'$ is statistcally close to e'). Hence the adversary's views in the games H_0, H_1 are statistically close.

Game H_2. Next, observe that no matter what the adversary does with its parts of the matrix A, the resulting $U = AR + \hat{E}$ is pseudo-random. This is because if we let R_j be the j'th block of n' rows in R then

$$U = \sum_{j \neq i} A_j R_j + (A_i R_i + \hat{E}).$$

and $(A_i R_i + \hat{E})$ is pseudorandom by LWE and independent of $\sum_{j \neq i} A_j R_j$.

In game H_2 we therefore replace the choise $U = AR + \hat{E}$ by a uniformly random $U \in \mathbb{Z}_q^{(m-1) \times w}$, and by the above H_2 is indistinguishable from H_1 under LWE.

Game H_3. We replace $v = s_i U + e'$ by a uniform $v \in \mathbb{Z}_q^w$, thus getting the case $\sigma = 0$.

We note that s_i has a lot of min-entropy left even given $b_i = s_i A$, since b_i leaks at most $n \log q$ bits about s_i (and we set m much larger than that). By leakage-resilience of LWE [GKPV10], the vector $s_i U + e'$ is pseudorandom when U is random and s_i has sufficient min-entropy.[2]

□

2.2 A Detour: The Need for Dual GSW

For the interested reader, we explain below why we need to use the "dual" GSW scheme rather than the "primal" GSW as in [CM15, MW16]. As we explained, the main differece between the primal and dual schemes is that that the matrix A in "primal" GSW is $(n-1)$-by-m, while in our "dual" scheme it is $(m-1)$-by-n (in both cases we have $m < n \log q$). While it is certailny possible that a one-round ILWE-hard protocol exists also for the "primal" scheme, we were not able to find one that we can prove secure under any standard assumption. Below we detail some specific failed attempts.

Failed attempt #1, parties choose different columns. Consider a protocol simlar to the one in Section 2.1.3, in which each party P_i is choosing a random $n \times m'$ matrix A_i and the matrix $A \in \mathbb{Z}_q^{n \times m'N}$ is just the column-concatenation of all the A_i's, $A = (A_1|A_2|\ldots|A_N)$.

An adversary (who controls P_N without loss of generality), can just set its matrix as $A_N = G$ where G is the GSW "gadget matrix". That gadget matrix has the property that given the vector $sG + e$ for a small error vector e, it is easy to find e and s. Now, notice that the vector $sA + e$ that P_N sends to $\mathcal{A}$ has the form $(sA_1 + e_1|sA_2 + e_2|\ldots|sA_N + e_N)$, so in particular the adversary can set the portion $sA_N + e_N = sG + e_N$ to recover the secret key s. (This is exactly where the "dual" scheme helps: the adversary still sees some "leakage" $sA_N + e_N$, but it cannot recover s since s still has a lot of min-entropy even given that leakage.)

Failed attempt #2, parties choose different rows. One way to avoid attacks as above is to ensure that for any fixed matrix that the adversary may put in "its entries", a random matrix by the honest user will make $sA + e$ pseudorandom.

One way to ensure this is to let each party choose a random $n' \times m$ matrix A_i and set $A \in \mathbb{Z}_q^{Nn' \times m}$ as the row-concatenation of the A_i's, i.e., $A^T = (A_1^T|\ldots|A_N^T)$. It is now easy to prove that $sA + e$ is pseudorandom (under LWE), no matter what the adversary does. But this arrangement opens another avenue of attack: The adversary (still controlling P_N) set $A_N = A_1$, so the bottom few rows in A are equal to the top few rows. Hence, also the

[2] Note that the instance $v = s_i U + e'$ is revealed only after the matrix A (and hence the leakage function on s_i) is set, so we do not need to deal with "after-the-fact" leakage.

bottom few rows in AR are equal to the top few rows, which lets the adversary distinguish AR from a uniform random U.

Some other failed attempts. At this point one may hope that if we let the parties choose different diagonals then neither of the attacks above would apply, but this is not the case. Here too, an adversary controlling all but one party can force the matrix A to have many identical rows, which would mean that so does the matrix AR. More generally, it seems that any arrangement where each party chooses a subset of the entries in A will let the adversary force A to be low rank, and hence also AR will be of low rank. (Here too the "dual" scheme works better, since the attacker sees $AR + E$ rather than AR itself.)

3 Other Preliminiaries

3.1 Commitment Schemes

Commitment schemes allow a *committer* C to commit itself to a value while keeping it (temporarily) secret from the *receiver* R. Later the commitment can be "opened", allowing the receiver to see the committed value and check that it is consistent with the earlier commitment. In this work, we consider commitment schemes with *statistically binding*. This means that even an unbounded cheating committer cannot create a commitment that can be opened in two different ways. We also use *tag-based* commitment, which means that in addition to the secret committed value there is also a public tag associated with the commitment. The notion of hiding that we use is *adaptive-security* (due to Pandey et al. [PPV08]): it roughly means that the committed value relative to some tag is hidden, even in a world that the receiver has access to an oracle that breaks the commitment relative to any other tag.

Definition 1 (Adaptively-secure Commitment[PPV08])**.** A tag-based commitment scheme (C, R) is statistically binding and adaptively hiding if it satisfies the following properties:

Statistical binding: For any (computationally unbounded) cheating committer C^* and auxiliary input z, it holds that the probability after the commitment stage that there exist two executions of the opening stage in which the receiver outputs two different values (other than $\perp$), is negligible.

Adaptive hiding: For every cheating PPT receiver R^* and every tag value tag, it holds that the following ensembles are computationally indistinguishable.

- $\{\mathsf{view}_{\mathsf{bCom}}^{R^*(\mathsf{tag}),\mathcal{B}_{\mathsf{tag}}}(m_1, z)\}_{\kappa\in N, m_1,m_2\in\{0,1\}^\kappa, z\in\{0,1\}^*}$
- $\{\mathsf{view}_{\mathsf{bCom}}^{R^*(\mathsf{tag}),\mathcal{B}_{\mathsf{tag}}}(m_2, z)\}_{\kappa\in N, m_1,m_2\in\{0,1\}^\kappa, z\in\{0,1\}^*}$

where $\mathsf{view}_{\mathsf{bCom}}^{R^*(\mathsf{tag}),\mathcal{B}_{\mathsf{tag}}}(m,z)$ denotes the random variable describing the output of $R^*(\mathsf{tag})$ after receiving a commitment to m relative to tag using bCom, while interacting with a commitment-breaking oracle $\mathcal{B}_{\mathsf{tag}}$.

The oracle $\mathcal{B}_{\mathsf{tag}}$ gets as input an alleged view v' and tag tag'. If $\mathsf{tag}' \neq \mathsf{tag}$ and v' is a valid transcript of a commitment to some value m' relative to tag', then $\mathcal{B}_{\mathsf{tag}}$ returns that value m'. (If there is no such value, or if $\mathsf{tag} = \mathsf{tag}'$, then $\mathcal{B}'_{\mathsf{tag}}$ returns $\perp$. If there is more than one possible value m' then $\mathcal{B}_{\mathsf{tag}'}$ returns an arbitrary one.)

To set up some notations, for a two-message commitment we let $\mathsf{com}_1 = \mathsf{bCom}_{\mathsf{tag}}(r)$ and $\mathsf{com}_2 = \mathsf{bCom}_{\mathsf{tag}}(m; \mathsf{com}_1; r')$ denote the two messages of the protocol, the first depending only on the randomness of the receiver and the second depending on the message to be committed, the first-round message from the receiver, and the randomness of the sender.

Pandey et al. [PPV08] proved that adaptively secure commitments exist if adaptive PRGs exist. Note that adaptive security implies non-malleability (which is the "intuitive nontion that we need"), but the other direction is not known. In our proof we rely heavily on adaptive security, we do not know if similar result can be proven based on any (two-round) non-malleable commitment.

3.2 Interactive Proofs

Given a pair of interactive Turing machines, P and V , we denote by $\langle P(w), V\rangle(x)$ the random variable representing the (local) output of V, on common input x, when interacting with machine P with private input w, when the random input to each machine is uniformly and independently chosen.

Definition 2 (Interactive Proof System)**.** A pair of interactive machines $\langle P, V\rangle$ is called an *interactive proof system* for a language L if there is a negligible function $\mu(\cdot)$ such that the following two conditions hold:

- Completeness: For every $x \in L$, and every $w \in R_L(x)$, $Pr[\langle P(w), V\rangle(x) = 1] = 1$.
- Soundness: For every $x \notin L$, and every interactive machine P^*, $Pr[\langle P^*, V\rangle(x) = 1] \leq \mu(\kappa)$

In case the soundness condition is required to hold only with respect to a computationally bounded prover, the pair $\langle P, V\rangle$ is called an interactive argument system.

3.3 Zero-Knowledge

We recall the standard definition of ZK proofs. Loosely speaking, an interactive proof is said to be zero-knowledge (ZK) if a verifier V learns nothing beyond the validity of the assertion being proved, it could not have generated on its own. As "feasible"computation in general is defined though the notion of probabilistic polynomial-time, this notion is

formalized by requiring that the output of every (possibly malicious) verifier interacting with the honest prover P can be "simulated" by a probabilistic expected polynomial-time machine $\mathcal{S}$ (a.k.a. the simulator). The idea behind this definition is that whatever V^* might have learned from interacting with P, he could have learned by himself by running the simulator $\mathcal{S}$.

Definition 3 (ZK)**.** . Let L be a language in $\mathcal{NP}$, R_L a witness relation for L, (P, V) an interactive proof (argument) system for L. We say that (P, V) is *statistical/computational ZK*, if for every probabilistic polynomial-time interactive machine V there exists a probabilistic algorithm $\mathcal{S}$ whose expected running-time is polynomial in the length of its first input, such that the following ensembles are statistically close/computationally indistinguishable over L.

- $\{\langle P(y), V(z)\rangle(x)\}_{\kappa\in\mathbb{N}\, x\in\{0,1\}^\kappa\cap L, y\in R_L(x), z\in\{0,1\}^*}$
- $\{\mathcal{S}(x, z)\}_{\kappa\in\mathbb{N}\, x\in\{0,1\}^\kappa\cap L, y\in R_L(x), z\in\{0,1\}^*}$

where $\langle P(y), V(z)\rangle(x)$ denotes the view of V in interaction with P on common input x and private inputs y and z, respectively.

3.4 Witness Indistinguishability

An interactive proof (or argument) is said to be witness indistinguishable (WI) if the verifier's output is "computationally" independent of the witness used by the prover for proving the statement. In this context, we focus on languages $L \in \mathcal{NP}$ with a corresponding witness relation R_L. Namely, we consider interactions in which, on common input x, the prover is given a witness in $R_L(x)$. By saying that the output is computationally independent of the witness, we mean that for any two possible $\mathcal{NP}$-witnesses that could be used by the prover to prove the statement $x \in L$, the corresponding outputs are computationally indistinguishable.

Definition 4 (Witness-indistinguishability)**.** . Let $\langle P, V\rangle$ be an interactive proof (or argument) system for a language $L \in \mathcal{NP}$. We say that $\langle P, V\rangle$ is *witness-indistinguishable* for R_L, if for every probabilistic polynomial-time interactive machine V^* and for every two sequences $\{w^1_{\kappa,x}\}_{\kappa\in\mathbb{N}, x\in L}$ and $\{w^2_{\kappa,x}\}_{\kappa\in\mathbb{N}, x\in L}$, such that $w^1_{\kappa,x}, w^2_{\kappa,x} \in R_L(x)$ for every $x \in L \cap \{0,1\}^\kappa$, the following probability ensembles are computationally indistinguishable over $\kappa \in \mathbb{N}$.

- $\{\langle P(w^1_{\kappa,x}), V^*(z)\rangle(x)\}_{\kappa\in\mathbb{N}\, x\in\{0,1\}^\kappa\cap L, z\in\{0,1\}^*}$
- $\{\langle P(w^2_{\kappa,x}), V^*(z)\rangle(x)\}_{\kappa\in\mathbb{N}\, x\in\{0,1\}^\kappa\cap L, z\in\{0,1\}^*}$

3.5 Proofs (Arguments) of Knowledge

Loosely speaking, an interactive proof is a proof of knowledge if the prover convinces the verifier that it possesses, or can feasibly compute, a witness for the statement proved. The notion of a proof of knowledge is essentially formalized as follows: an interactive proof of $x \in L$ is a proof of knowledge if there exists a probabilistic expected polynomial-time extractor machine E, such that for any prover P, E on input the description of P and any statement $x \in L$ readily outputs a valid witness for $x \in L$ if P succeeds in convincing the Verifier that $x \in L$. Formally,

Definition 5 (Proof of knowledge)**.** Let (P, V) be an interactive proof system for the language L. We say that (P, V) is a proof of knowledge for the witness relation R_L for the language L it there exists an probabilistic expected polynomial-time machine E, called the extractor, and a negligible function $\mu(\cdot)$ such that for every machine P^*, every statement $x \in \{0,1\}^\kappa$, every random tape $x \in \{0,1\}^*$, and every auxiliary input $z \in \{0,1\}^*$,

$$Pr[\langle P_r^*(z), V\rangle(x) = 1] \leq Pr[E^{P_r^*(x,z)}(x) \in R_L(x)] + \mu(\kappa)$$

An interactive argument system $\langle P, V\rangle$ is an argument of knowledge if the above condition holds w.r.t. probabilistic polynomial-time provers.

Delayed-Input Witness Indistinguishability. The notion of delayed-input Witness Indistinguishability formalizes security of the prover with respect to an adversarial verifier that adaptively chooses the input statement to the proof system in the last round. Once we consider such adaptive instance selection, we also need to specify where the witnesses come from; to make the definition as general as possible, we consider an arbitrary (potentially unbounded) *witness selecting machine* that receives as input the views of all parties and outputs a witness w for any statement x requested by the adversary. In particular, this machine is a (randomized) Turing machine that runs in exponential time, and on input a statement x and the current view of all parties, picks a witness $w \in R_L(x)$ as the private input of the prover.

Let $\langle P, V\rangle$ be a 3-round Witness Indistinguishable proof system for a language $L \in \mathcal{NP}$ with witness relation R_L. Denote the messages exchanged by $(\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3)$ where p_i denotes the message in the i-th round. For a delayed-input 3-round Witness Indistinguishable proof system, we consider the game ExpAWI between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ in which the instance x is chosen by $\mathcal{A}$ after seeing the first message of the protocol played by the challenger. Then, the challenger receives as local input two witnesses w_0 and w_1 for x chosen adaptively by a witness-selecting machine. The challenger then continues the game by randomly selecting one of the two witnesses and by computing the third message by running the prover's algorithm on input the instance x, the selected witness w_b and the challenge received from the adversary in the second round. The adversary wins the game if he can guess which of the two witnesses was used by the challenger.

Definition 6 (Delayed-Input Witness Indistinguishability)**.** Let $\mathsf{ExpAWI}^{\mathcal{A}}_{\langle P,V\rangle}$ be a delayed-input WI experiment parameterized by a PPT adversary $\mathcal{A}$ and an delayed-input 3-round Witness Indistinguishable proof system $\langle P, V\rangle$ for a language $L \in \mathcal{NP}$ with witness relation R_L. The experiment has as input the security parameter κ and auxiliary information aux for $\mathcal{A}$. The experiment ExpAWI proceeds as follows:

$\mathsf{ExpAWI}^{\mathcal{A}}_{\langle P,V\rangle}(\kappa, aux)$:

Round-1: The challenger $\mathcal{C}$ randomly selects coin tosses r and runs P on input $(1^\kappa; r)$ to obtain the first message p_1;

Round-2: $\mathcal{A}$ on input p_1 and aux chooses an instance x and a challenge p_2. The witness-selecting machine on inputs the statement x and the current view of all parties outputs witnesses w_0 and w_1 such that $(x, w_0), (x, w_1) \in R_L$. $\mathcal{A}$ outputs $x, w_0, w_1, \mathsf{p}_2$ and internal state state;

Round-3: $\mathcal{C}$ randomly selects $b \leftarrow \{0, 1\}$ and runs P on input (x, w_b, p_2) to obtain p_3;

$b' \leftarrow \mathcal{A}((\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3), aux, \mathsf{state})$;

If $b = b'$ then output 1 else output 0.

A 3-round Witness Indistinguishable proof system for a language $L \in \mathcal{NP}$ with witness relation R_L is *delayed-input* if for any PPT adversary $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that for any $aux \in \{0, 1\}^*$ it holds that

$$|Pr[\mathsf{ExpAWI}^{\mathcal{A}}_{\langle P,V\rangle}(\kappa, aux) = 1] - 1/2| \leq \mu(\kappa)$$

The most recent 3-round delayed-input WI proof system appeared in the work of [COSV16].

3.6 Feige-Shamir ZK Proof Systems

For our construction we use the 3-round, public-coin, input-delayed witness-indistinguishable proof-of-knowledge Π_{WIPOK} based on the work of Feige, Lapidot, Shamir [FLS99], and the 4-round zero-knowledge argument-of-knowledge protocol of Feige and Shamir Π_{FS} [FS90].

Recall that the Feige-Shamir protocol consists of two executions of a WIPOK protocol in reverse directions. The first execution has the verifier prove something about a secret that it chooses, and the second execution has the prover proving that either the input statement is true or the prover knows the verifier's secret. The zero-knowledge simulator then uses the knowledge extraction to extract the secret of the verifier, making it possible to complete the proof.

4 Multi-Party Computation Protocol

For our protocol we use the following components:

- The "dual"-GSW-based scheme for simulatable threshold multi-key FHE from Section 2, $\mathsf{MFHE} = (\mathsf{FHE.Setup}, \mathsf{MFHE.Keygen}, \mathsf{MFHE.Encrypt}, \mathrm{MFHE.Eval}, \mathsf{MFHE.PartDec}, \mathsf{MFHE.FinDec})$, with its one-round initialization protocol $\Pi_{\mathsf{GenSetup}} = (\mathsf{Gen}_1, \mathsf{Gen}_2)$ for computing the matrix A.
- Two instances of a two-round adaptively secure commitment scheme, supporting tags/identities of length κ. We denote the first instance by $\mathsf{aCom} = (\mathsf{acom}_1, \mathsf{acom}_2)$ and the second by $\mathsf{bCom} = (\mathsf{bcom}_1, \mathsf{bcom}_2)$.[3]
- A one-way function OWF.
- A three-round public coin witness-indistinguishable proof of knowledge $\Pi_{\mathrm{WIPOK}} = (\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3)$ for the $\mathcal{NP}$-Language $\mathcal{L}_P^{\mathrm{WIPOK}}$ where party P acts as the Prover, with dealyed input (where the statement is decided in the last round);
- A four-round zero-knowledge argument of knowledge $\Pi_{\mathrm{FS}} = (\mathsf{fs}_1, \mathsf{fs}_2, \mathsf{fs}_3, \mathsf{fs}_4)$ for the $\mathcal{NP}$-Language $\mathcal{L}_P^{\mathrm{FS}}$ where party P acts as the Prover, with delayed input.

The protocol. Let $F : \{0,1\}^{\kappa \times N} \to \{0,1\}^{\kappa}$ be a deterministic function to be computed. Each party P_i holds input $x_i \in \{0,1\}^{\kappa}$ and identity id_i. (We note that known transformations yield a protocol for randomized functionalities, without increasing the rounds, see [Gol04, Section 7.3].) The protocol consists of four broadcast rounds, where messages $(m_t^1, \ldots, m_t^N)$ are exchanged simultaneously in the t-th round for $t \in [4]$. The message flow is detailed in Figure 1, and Figure 3 depicts the exchanged messages between two parties P_i and P_j.

4.1 Proof of Security

Theorem 2. Assuming sub-exponential hardness of LWE, and the existence of an adaptively-secure commitment scheme, there exists a four-broadcast-round protocol for securely realizing any functionality against malicious adversary in the plain model with no setup.

To prove Theorem 2, we note that the two assumptions listed suffice for instantiating all the components of our protocol Π_{MPC}: The commitment is used directly for aCom and bCom, and sub-exponential LWE suffices for everything else. Below we prove security of Π_{MPC} by describing a simulator and proving that the simulated view is indistinguishable from the real one.

[3] Strictly speaking we do not need the second instance to be adaptively secure, but it is convenient to use the same scheme for both instances.

4.1.1 Description of the Simulator

Let $\mathcal{A}$ be a malicious, static adversary that interacts with parties running the protocol Π_{MPC} from Figure 4 in the plain model. We construct a simulator $\mathcal{S}$ (the ideal world adversary) with access to the ideal functionality $\mathcal{F}$, which simulates a real execution of Π_{MPC} with $\mathcal{A}$ such that the ideal world experiment with $\mathcal{S}$ and $\mathcal{F}$ is indistinguishable from a real execution of Π_{MPC} with $\mathcal{A}$. Our simulator $\mathcal{S}$ proceeds as follows:

Simulating actual protocol messages in Π: Let $\mathcal{P} = \{P_1, \ldots, P_N\}$ be the set of parties participating in the execution of Π_{MPC}. Also let $\mathcal{P}^* \subseteq \mathcal{P}$ be the set of parties corrupted by the adversary $\mathcal{A}$. The simulator $\mathcal{S}$ only generates messages on behalf of parties $\mathcal{P} \backslash \mathcal{P}^*$.

Round 1 Messages $\mathcal{S} \to \mathcal{A}$: In the first round $\mathcal{S}$ generates messages on behalf of each honest party $P_h \notin \mathcal{P}^*$, as follows:

1. Choose randomness $r_h = (r_h^{gen}, r_h^{enc})$ for the MFHE protocol and an unrelated κ-bit randomness value R_h, and set $\hat{R}_h = OWF(R_h)$.
2. For every j engage in a two-round commitment protocol with P_j. To this end, prepare the first message $\mathsf{acom}_1^{h,j}$ corresponding to the execution of $\mathsf{aCom}_{\mathsf{id}_j}(x_j, r_j^{gen}, r_j^{enc}, R_j; \omega_j)$ on behalf of P_h, acting as the receiver of the commitment. Since the commitment aCom is a two-round protocol, the message of the committer P_j is only sent in the second round.
3. Prepare the first message $\mathsf{p}_1^{h,j}$ of Π_{WIPOK}, where P_h acts as the Prover, for the $\mathcal{NP}$-Language $\mathcal{L}_{P_h}^{\text{WIPOK}}$ and the first message $\mathsf{fs}_1^{h,j}$ of Π_{FS} where P_h acts as the Verifier for $\mathcal{L}_{P_j}^{\text{FS}}$.
4. Honestly generate the message α_h of the 1-round protocol Π_{GenSetup}.
5. It then sends the message $m_1^{h,j} = \left(\hat{R}_h, \mathsf{acom}_1^{h,j}, \mathsf{p}_1^{h,j}, \mathsf{fs}_1^{h,j}, \alpha_h\right)$ to $\mathcal{A}$.

Round 1 Messages $\mathcal{A} \to \mathcal{S}$: Also in the first round the adversary $\mathcal{A}$ generates the messages $m_1^{j,h} = \left(\hat{R}_j, \mathsf{acom}_1^{j,h}, \mathsf{p}_1^{j,h}, \mathsf{fs}_1^{j,h}, \alpha_j\right)$ on behalf of corrupted parties $j \in \mathcal{P}^*$ to honest parties $h \notin \mathcal{P}^*$. Messages $\{\mathsf{acom}_1^{j,h}\}$ correspond to an execution of $\mathsf{aCom}_{\mathsf{id}_h}(\mathbf{0}; \omega_h)$.

Round 2 Messages $\mathcal{S} \to \mathcal{A}$: In the second round $\mathcal{S}$ generates messages on behalf of each honest party $P_h \in \mathcal{P}^*$ as follows:

1. Complete the commitment to the zero string generating the second messages $\mathsf{acom}_2^{j,h}$ corresponding to all executions of $\mathsf{aCom}_{\mathsf{id}_h}(\mathbf{0}; \omega_h)$.

2. Honestly prepare the second message $\mathsf{p}_2^{j,h}$ ($\mathsf{fs}_2^{j,h}$) of $\Pi_{\text{WIPOK}}(\Pi_{\text{FS}})$ initiated by P_j acting as the prover (verifier) in the first round.

3. Generate the second commitment messages $\mathsf{bcom}_1^{h,j}$ for $\mathsf{bCom}_{\mathsf{id}_j}(\mathbf{0};\zeta_j)$ where party P_h acts as the Receiver.

4. Generate the individual key pair by locally computing the matrix A (on input $\{\alpha_i\}_{i\in[N]}$), and by computing $(\mathsf{pk}_h, \mathsf{sk}_h) = \mathsf{MFHE.Keygen}(\mathsf{params}, A; r_h^{gen})$.

5. It then sends the message $m_2^{h,j} := (\mathsf{acom}_2^{j,h}, \mathsf{bcom}_1^{h,j}, \mathsf{p}_2^{j,h}, \mathsf{fs}_2^{j,h}, \mathsf{pk}_h)$ to $\mathcal{A}$.

Round 2 Messages $\mathcal{A} \to \mathcal{S}$: In the second round the adversary $\mathcal{A}$ generates the messages $m_2^{j,h} := (\mathsf{acom}_2^{h,j}, \mathsf{bcom}_1^{j,h}, \mathsf{p}_2^{h,j}, \mathsf{fs}_2^{h,j}, \mathsf{pk}_j)$ on behalf of corrupted parties $j \in \mathcal{P}^*$ to honest parties $h \notin \mathcal{P}^*$. Messages $\{\mathsf{acom}_2^{h,j}\}$ correspond to an execution of $\mathsf{aCom}_{\mathsf{id}_j}(x_j, r_j^{gen}, r_j^{enc}, R_j; \omega_j)$ and messages $\{\mathsf{bcom}_1^{j,h}\}$ correspond to an execution of $\mathsf{bCom}_{\mathsf{id}_h}(\mathbf{0};\zeta_h)$

Round 3 Messages $\mathcal{S} \to \mathcal{A}$: In the third round $\mathcal{S}$ generates messages on behalf of each honest party $P_h \notin \mathcal{P}^*$ as follows:

1. Generate the second messages $\mathsf{bcom}_2^{j,h}$ corresponding to all $\mathsf{bCom}_{\mathsf{id}_h}(\mathbf{0};\zeta_h)$.

2. Generate an encryption of the zero string using randomness r_h^{enc}, i.e. $c_h = \mathsf{MFHE.Encrypt}(pk_h, \mathbf{0}; r_h^{enc})$.

3. Honestly prepare the final message $\mathsf{p}_3^{h,j}$ ($\mathsf{fs}_3^{h,j}$) of $\Pi_{\text{WIPOK}}(\Pi_{\text{FS}})$ initiated by P_h acting as the prover (verifier) in the first round.

4. It sends the message $m_3^{h,j} = (\mathsf{bcom}_2^{j,h}, c_h, \mathsf{p}_3^{h,j}, \mathsf{fs}_3^{h,j})$ to $\mathcal{A}$.

Round 3 Messages $\mathcal{A} \to \mathcal{S}$: In the third round $\mathcal{A}$ generates $m_3^{j,h} = (\mathsf{bcom}_2^{h,j}, c_j, \mathsf{p}_3^{j,h}, \mathsf{fs}_3^{j,h})$ where messages $\{\mathsf{bcom}_2^{h,j}\}$ correspond to an execution of $\mathsf{bCom}_{\mathsf{id}_j}(\mathbf{0};\zeta_j)$. Then, $\mathcal{S}$ proceeds to extract the witness corresponding to each proof-of-knowledge completed in the first three rounds (via rewinding). To this end, $\mathcal{S}$ applies the knowledge extractor of Π_{WIPOK} to obtain the "witnesses" which consist of the inputs and secret keys of the corrupted parties (x_j, r_j)[4] and the zero knowledge simulator of Π_{FS} to obtain the "trapdoors" (which for a Feige-Shamir protocol means extracting the verifier-secret). If extraction fails, $\mathcal{S}$ outputs fail. Next $\mathcal{S}$ sends $\{x_j\}_{j\in[N]\setminus\{h\}}$ to the ideal functionality $\mathcal{F}$ which responds by sending back y such that $y = F(\{x_j\}_{j\in[N]})$.

[4]For simplicity of exposition, we omit the rest of the witness values.

Round 4 Messages $\mathcal{S} \to \mathcal{A}$: In the fourth round $\mathcal{S}$ generates messages on behalf of each honest party $P_h \notin \mathcal{P}^*$ as follows:

1. Generate the evaluated ciphertext $\hat{c} := \text{MFHE.Eval}(\mathsf{params}; F; (c_1, \ldots, c_N))$.

2. Then, $\mathcal{S}$ obtains all the secret keys $\{\mathsf{sk}_j\}_{j\in\mathcal{P}^*}$ reconstructed from the wintesses $\{r_j^{gen}\}_{j\in\mathcal{P}^*}$ and computes the simulated decryption shares $\{ev_h\}_{h\notin\mathcal{P}^*} \leftarrow \mathcal{S}^T(y, \hat{c}, h, \{\mathsf{sk}_j\}_{j\in\mathcal{P}^*})$. (The simulator $\mathcal{S}^T$ is the one provided by [MW16, Section 6.2].)

3. Fake the final message $\mathsf{fs}_4^{j,h}$ of Π_{FS} protocol using the extracted trapdoor. It sends the message $m_4^{h,j} = (ev_h, \mathsf{fs}_4^{j,h})$ on behalf of P_h.

Round 4 Messages $\mathcal{A} \to \mathcal{S}$: In the last round the adversary $\mathcal{A}$ generates the messages on behalf of corrupted parties in $\mathcal{P}^*$. For each party $j \in \mathcal{P}^*$ our simulator receives messages $m_4^{j,h} = (ev_j, \widetilde{\mathsf{fs}}_4^{h,j})$ from $\mathcal{A}$.

This completes the description of the simulator.

4.1.2 Proof of Indistinguishability

We need to prove that for any malicious (static) adversary $\mathcal{A}$, the view generated by the simulator $\mathcal{S}$ above is indistinguishable from the real view, namely:

$$\{\text{IDEAL}_{\mathcal{F},\mathcal{S}}(\kappa, \cdot)\}_\kappa \overset{\text{c}}{\approx} \{\text{REAL}_{\Pi,\mathcal{A}}(\kappa, \cdot)\}_\kappa$$

To prove indistinguishability, we consider a sequence of hybrid experiments $H_0, H_1, \ldots$ as described below. Let H_0 be the hybrid describing the real-world execution of the protocol. We modify this game in steps as follows:

- H_1 Use the zero-knowledge simulator to generate the proof in the 4-round Π_{FS}, indistinguishability follows by the ZK property of Π_{FS}.

- H_2 Starting in this hybrid, the challenger is given access to a breaking oracle $\mathcal{B}_{\mathsf{tag}}$ (with $\mathsf{tag} = (\mathsf{id}_h, \star)$ where h is a honest party). Here the challenger uses the breaking oracle to extract the values committed to by the adversary in $\mathsf{acom}_2^{h,\mathcal{A}}$ (in the second round), then commits to these same values in $\mathsf{bcom}_2^{\mathcal{A},h}$ on behalf of the honest party (in the third round). Indistinguishability follows by the adaptive-hiding of bCom.

- H_3 Change the proof in Π_{WIPOK} to use the "OR branch". Indistinguishability follows by the WI property of Π_{WIPOK} (which must hold even in the presence of the breaking-oracle $\mathcal{B}_{\mathsf{tag}}$).

H_4 Here the challenger also has access to the ideal-world functionality that gives it the output of the function. Having extracted the secret keys using $\mathcal{B}_{\mathsf{tag}}$, the challenger *simulates the decryption shares* of the honest parties rather than using the decryption procedure. Indistinguishability follows since the FHE scheme is simulatable (which follows from LWE).

H_5 Encrypt 0's rather than the true inputs. Indistinguishability follows due to the semantic security of the encryption scheme (that follows from the ILWE-hardness of our scheme, under LWE).

H_6 Commit to 0's in $\mathsf{acom}_2^{\mathcal{A},h}$, rather than to the real inputs. Indistinguishable due to the adaptive-hiding of aCom.

H_7 Revert the change in H_3, make the proof in Π_{WIPOK} use the normal branch rather than the "OR branch". Indistinguishability follows by the WI property of Π_{WIPOK}.

H_8 Revert the change in H_2 and thus commit to zero in $\mathsf{bcom}_2^{\mathcal{A},h}$ (instead of committing to the extracted values). Indistinguishability follows by the adaptive-hiding of bCom.

H_9 Here the challenger no longer has access to a breaking oracle, and instead it uses the POK extractor to get the randomness and inputs (witnesses) from Π_{WIPOK}. Indistinguishability follows from the extraction property of Π_{WIPOK}, combined with the one-wayness of OWF.

As H_9 we no longer uses the inputs of the honest parties, the view of this hybrid can be simulated. (We also note that the simulator *does not use a breaking oracle*, rather it is a traditional rewinding simulator.)

Security in the presence of a breaking oracle: Note that some of our indistinguishability arguments must holds in worlds with a breaking oracle $\mathcal{B}_{\mathsf{tag}}$. In particular, we require that bCom is still hiding, that LWE still holds, and that Π_{WIPOK} is still witness-indistinguishable in the presence of the oracle. The hiding property of bCom follows directly from its adaptive-hiding property. As for LWE and Π_{WIPOK}, security in the presence of $\mathcal{B}_{\mathsf{tag}}$ follows from sub-exponential hardness and complexity leveraging. Namely, in the relevant reductions we can implement $\mathcal{B}_{\mathsf{tag}}$ ourselves in subexponential time, while still relying on the hardness of LWE or Π_{WIPOK}.

Another point to note is that using the zero-knowledge simulator (in hybrids H_2-H_9) requires rewinding, which may be problematic when doing other reductions. As we explain below, we are able to handle rewinding by introducing many sub-hybrids, essentially cutting the distinguishing advantage by a factor equals to the number of rewinding operations.

$\mathbf{H_0}$: This hybrid is the real execution. In particular, H_0 starts the execution of $\mathcal{A}$ providing it fresh randomness and input $\{x_j\}_{P_j \in \mathcal{P}^*}$, and interacts with it honestly by performing

all actions of the honest parties with uniform randomness and input. The output consists of $\mathcal{A}$'s view.

$\mathbf{H_1}$: In this hybrid the challenger uses the zero-knowledge simulator of Π_{FS} to generate the proofs on behalf of each honest party P_h, rather than the honest prover strategy as is done in $\mathbf{H_0}$. We note that the challenger in this hybrid needs to rewind the adversary $\mathcal{A}$ (up to the second round) as needed for the Feige-Shamir ZK simulator.

Since in these two hybrids the protocol Π_{FS} is used to prove the same true statement, then the simulated proofs are indistinguishable from the real ones, so we get:

Lemma 4.1. $\text{H}_0 \approx_{\text{s}} \text{H}_1$.

$\mathbf{H_2}$: In this "mental-experiment hybrid" the challenger is given access to a breaking oracle $\mathcal{B}_{\mathsf{id}_h}$, with the tag being the identity of an arbitrary honest parties ($h \notin \mathcal{P}^*$). The challenger begins as in the real execution for the first two rounds, but then it uses $\mathcal{B}_{\mathsf{tag}}$ to extract the values (x_j, r_j, R_j) of all the adversarial players $j \in \mathcal{P}^*$ from $\mathsf{acom}_2^{h,j}$.

Then the challenger changes the commitments $\mathsf{bcom}_2^{j,h}$ on behalf of the honest party P_h, committing to the values R_j that were extracted from $\mathsf{acom}_2^{h,j}$ (and thus making the language $\mathcal{L}_{h,j,2}$ –the "OR branch"– in Π_{WIPOK} a true statement).[5]

Lemma 4.2. $\text{H}_1 \approx_{\text{c}} \text{H}_2$.

Proof: Since the only differences between these hybrids are the values committed to in $\mathsf{bcom}^{j,h}$, then indistinguishability should follow from the adaptive-hiding of the commitment scheme bCom (as the challenger never queries its breaking oracle with any tag containing the identity id_h of the honest party).

One subtle point here, is that in both H_1 and H_2 we use the rewinding Feige-Shamir ZK simulator, so we need to explain how the single value $\mathsf{bcom}_2^{j,h}$ provided by the committer in the reduction (which is a commitment to either 0 or R_j) is used in all these transcripts. To that end let M be some polynomial upper bound on the number of rewinding operations needed by the zero-knowledge simulator. The reduction to the security of bCom will choose at random $t \in [1, M]$ and will only use the bCom committer that it interacts with to commit to a value in the t'th rewinding, committing to 0 in all the rewindings $i < t$ and to the value R_j (that it has from the breaking oracle) in all the rewindings $i > t$.

By a standard argument, if we can distinguish between $\text{H}_1 \approx_{\text{c}} \text{H}_2$ with probability ϵ then the reduction algorithm can distinguish commitments to 0 and R_j with probability ϵ/M. □

[5] The commitment *Com* starts in the second round, but this is a two-round commitment so the committed value only affects the second message in the commitment, which happens in the third round of the larger protocol.

$\mathbf{H_3}$: In this hybrid, we change the witness used in Π_{WIPOK} on behalf of each honest party P_h. In particular, all Π_{WIPOK} executions use the "OR branch" $\mathcal{L}_{h,j,2}$.

Lemma 4.3. $\text{H}_2 \approx_{\text{c}} \text{H}_3$.

Proof: We make sub-hybrids that change one honest party at a time, and show that a distinguisher D that distinguishes two such sub-hybrids can be used by another distinguisher D' to distinguish between the two witnesses of Π_{WIPOK} (as per Definition 6).

Description of D': D' plays the role of both the challenger and the adversary in the two hybrids, except that the prover messages of Π_{WIPOK} (on behalf of P_h) are obtained from the external prover that the WI-distinguisher D' has access to.

At the third round of the protocol, D' has the statement that P_h needs to prove, and it gets the two witnesses for that statement from the witness-selecting machine in Definition 6. Sending the statement and witnesses to its external prover, D' obtains the relevant Π_{WIPOK} message (for one of them). D' also uses these witnesses to complete the other flows of the protocol (e.g., the commitments $\mathsf{bcom}_2^{j,h}$ that include some of these witnesses). Once the protocol run is finished, it gives the transcript to D and outputs whatever D outputs.

As above, we still need to support rewinding by the Feige-Shamir ZK simulator, while having access to only a single interaction with the external prover, and we do it by sub-sub-hybrids where we embed this interaction in a random rewinding t, producing all the other proofs by the H_2 challenger (for $i < t$) or the H_3 challenger (for $i > t$). It is clear that the advantage of D' is a $1/M$ fraction of the advantage of D. □

We note that D' above still uses the breaking oracle $\mathcal{B}_{\mathsf{tag}}$ (to extract the Π_{FS} secrets), so we need to assume that delayed-input-WI holds even in a world with the breaking oracle. As explained above, we rely on complexity leveraging for that purpose. That is, we let D' run in subexponential time (so it can implement $\mathcal{B}_{\mathsf{tag}}$ itself), and set the parameters of Π_{WIPOK} large enough so we can assume witness-indistinguishability even for such a strong D'. (We can implement subexponential WI protocol from subexponential LWE.)

$\mathbf{H_4}$: The difference from H_3 is that in H_4 we simulate the decryption shares of the honest parties. More specifically, the challenger in H_4 has access also to the ideal functionality, and it proceeds as follows:

1. It completes the first three broadcast rounds exactly as in H_3.
2. Having extracted the input of all the corrupted parties, the challenger sends all these inputs to the ideal functionality $\mathcal{F}$ and receives back the output $y = F(\{x_j\}_{j\in[N]})$.

3. Having extracted also all the secret keys of the corrupted parties, the challenger has everything that it needs to compute the simulated decryption shares of the honest parties, $\{ev_h\}_{h\notin\mathcal{P}^*} \leftarrow \mathcal{S}^T(y, \hat{c}, h, \{\mathsf{sk}_j\}_{j\in\mathcal{P}^*})$.
4. The challenger computes also the last message of Π_{FS} (using the simulator as before), and sends it together with decryption shares $\{ev_h\}_h$ in the last round.

Lemma 4.4. $\text{H}_3 \approx_{\text{s}} \text{H}_4$.

Proof: The only change between these two experiments is that the partial decryption shares of the honest parties are not generated by partial decryption. Instead they are generated via the the threshold simulator $\mathcal{S}^T$ of the MFHE scheme. By the simulatability of threshold decryption, the partial decryptions shares are statistically indistinguishable. □

H_5: We change H_4 by making $\mathcal{S}$ broadcast encryptions of $\mathbf{0}$ on behalf of the honest parties in the third round, instead of encrypting the real inputs.

Lemma 4.5. $\text{H}_4 \approx_{\text{c}} \text{H}_5$.

Proof: The proof follows directly from semantic security, which in our case follows from LWE. As in the previous hybrid, here too we need this assumption to hold even in the presence of a breaking oracle, and we lose a factor of M in the distinguishing probability due to rewinding. □

H_6: In this hybrid, we get rid of the honest partys' inputs $\{(x_h, r_h)\}_h$ (that are present in the values of $\mathsf{acom}_2^{j,h}$). Formally, H_6 is identical to H_5 except that in the first round it sets $x_h = \mathbf{0}$ for all $h \notin \mathcal{P}^*$.

Lemma 4.6. $\text{H}_5 \approx_{\text{c}} \text{H}_6$.

Proof: This proof is very similar to the the proof of $\text{H}_1 \approx_{\text{c}} \text{H}_2$, and indistinguishability follows from adaptive-hiding of aCom. Since the challenger never asks its breaking oracle $\mathcal{B}_{\mathsf{tag}}$ to break commitments relative to the honest party's tags (and since these committed values are no longer used by the challenger for anything else), then having the honest parties commit to x_h is indistinguishable from having it commit to $\mathbf{0}$. □

H_7: In this hybrid we essentially reverse the change that was made in going from H_2 to H_3. Namely, since now both the encryption and the commitment at each honest party are for the value $\mathbf{0}$ then there is no need to use the "OR branch" in Π_{WIPOK}. Hence we return in using the honest prover strategy there, relative to the input $x_h = 0$. As in Lemma 4.3 indistinguishability follows by the WI property of Π_{WIPOK}.

$\mathbf{H_8}$: Revert the change that was made in going from H_1 to H_2 and thus commit to a random value s_h in $\mathsf{bcom}_2^{j,h}$. Indistinguishability follows by the computational hiding of bCom, just like in Lemma 4.2.

$\mathbf{H_9}$: In this hybrid the challenger no longer has access to the breaking oracle $\mathcal{B}_{\mathsf{tag}}$. Instead, it uses the knowledge extractor of Π_{WIPOK} to get the input and secret keys of the corrupted parties, and the "standard" zero-knowledge simulator to get the proof in Π_{FS}.

Lemma 4.7. $\text{H}_8 \approx_s \text{H}_9$.

Proof: The only difference between these hybrids is the method used by the challenger to extract the adversary secrets. Two technical points needs to be addressed here:

- This hybrid requires rewinding by *both* the FS ZK simulator and the FLS knowledge extractor, so we need to argue that after polynomially many trials they will *both succeed* on the same transcript. This is a rather standard argument (which essentially boils down to looking at the knowledge-extractor inside Π_{FS} and the one used explicitly in Π_{WIPOK} as extracting knowledge for and AND language.)
- We also need to argue that the value extracted from the adversary by the Π_{WIPOK} extractor in H_9 is a witness for $\mathcal{L}_{i,j,1}$ and not for $\mathcal{L}_{i,j,2}$. This is done by appealing to the one-wayness of OWF, if there is a noticeable probability to extract an $\mathcal{L}_{i,j,2}$ witness in H_9 then we get an inverter for this one-way function.

We conclude that in both H_8 and H_9 we succeed in extraction with about the same probability, and moreover extract the very same thing, and (statistical) indistinguishability follows. □

We conclude the proof by observing that the hybrid H_9 is identical to the ideal-world game with the simulator. □

References

[AIK05] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 260–274, 2005.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval

and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd FOCS*, pages 106–115. IEEE Computer Society Press, October 2001.

[BD10] Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 201–218. Springer, Heidelberg, February 2010.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

[BL04] Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. *SIAM J. Comput.*, 33(4):738–818, 2004.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990.

[BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key FHE with short ciphertexts. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 190–213. Springer, Heidelberg, August 2016.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 630–656. Springer, Heidelberg, August 2015.

[COSV16] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Concurrent non-malleable commitments (and more) in 3 rounds. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 270–299. Springer, Heidelberg, August 2016.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 378–394. Springer, Heidelberg, August 2005.

[DI06] Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 501–520. Springer, Heidelberg, August 2006.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1):1–28, 1999.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd ACM STOC*, pages 416–426. ACM Press, May 1990.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[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. Springer, Heidelberg, February 2014.

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS*, pages 230–240. Tsinghua University Press, 2010.

[GLOV12] Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd FOCS*, pages 51–60. IEEE Computer Society Press, October 2012.

[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. Springer, Heidelberg, 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 *STOC*, pages 218–229, 1987.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 695–704. ACM Press, June 2011.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

[HK07] Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 111–129. Springer, Heidelberg, August 2007.

[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. Springer, Heidelberg, August 2011.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st FOCS*, pages 294–304. IEEE Computer Society Press, November 2000.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008.

[JS07] Stanislaw Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 97–114. Springer, Heidelberg, May 2007.

[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. Springer, Heidelberg, August 2004.

[KOS03] Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Round efficiency of multi-party computation with a dishonest majority. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 578–595. Springer, Heidelberg, May 2003.

[Lin01] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 171–189. Springer, Heidelberg, August 2001.

[LP11a] Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 705–714. ACM Press, June 2011.

[LP11b] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 329–346. Springer, Heidelberg, March 2011.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, 2012. Full version at http://ia.cr/2011/501.

[MSS11] Steven Myers, Mona Sergi, and Abhi Shelat. Threshold fully homomorphic encryption and secure computation. Cryptology ePrint Archive, Report 2011/454, 2011. http://eprint.iacr.org/2011/454.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In *EUROCRYPT (2)*, volume 9666 of *Lecture Notes in Computer Science*, pages 735–763. Springer, 2016. Full version at http://ia.cr/2015/345.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[ORS15] Rafail Ostrovsky, Silas Richelson, and Alessandra Scafuro. Round-optimal black-box two-party computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 339–358. Springer, Heidelberg, August 2015.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, *36th ACM STOC*, pages 232–241. ACM Press, June 2004.

[PPV08] Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 57–74. Springer, Heidelberg, August 2008.

[PS16] Chris Peikert and Sina Shiehian. Multi-key FHE from LWE, revisited. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 217–238. Springer, Heidelberg, October / November 2016.

[Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *51st FOCS*, pages 531–540. IEEE Computer Society Press, October 2010.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

[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.

A Secure Computation Definitions

For completeness, we recall the definition of secure computation based on [Gol04, Chapter 7] here. We only recall the two party case as it is most relevant to our proofs. The description naturally extends to multi-party case as well (details can be found in [Gol04]).

Two-party computation. A two-party protocol problem is cast by specifying a random process that maps pairs of inputs to pairs of outputs (one for each party). We refer to such a process as a functionality and denote it $\mathcal{F} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ where $\mathcal{F} = (F_1, F_2)$. That is, for every pair of inputs (x, y), the output-pair is a random variable $(F_1(x, y), F_2(x, y))$ ranging over pairs of strings. The first party (with input x) wishes to obtain $F_1(x, y)$ and the second party (with input y) wishes to obtain $F_2(x, y)$.

Adversarial behavior. Loosely speaking, the aim of a secure two-party protocol is to protect an honest party against dishonest behavior by the other party. In this paper, we consider malicious adversaries who may arbitrarily deviate from the specified protocol. When considering malicious adversaries, there are certain undesirable actions that cannot be prevented. Specifically, a party may refuse to participate in the protocol, may substitute its local input (and use instead a different input) and may abort the protocol prematurely. One ramification of the adversary's ability to abort, is that it is impossible to achieve *fairness*. That is, the adversary may obtain its output while the honest party does not. In this work we consider a static corruption model, where one of the parties is adversarial and the other is honest, and this is fixed before the execution begins.

Communication channel. In our results we consider a secure simultaneous message exchange channel in which all parties can simultaneously send messages over the channel at the same communication round. Moreover, we assume an asynchronous network[6] where

[6] The fact that the network is asynchronous means that the messages are not necessarily delivered in the order which they are sent.

the communication is open (i.e. all the communication between the parties is seen by the adversary) and delivery of messages is not guaranteed. For simplicity, we assume that the delivered messages are authenticated. This can be achieved using standard methods.

Security of protocols (informal). The security of a protocol is analyzed by comparing what an adversary can do in the protocol to what it can do in an ideal scenario that is secure by definition. This is formalized by considering an ideal computation involving an incorruptible trusted third party to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. Loosely speaking, a protocol is secure if any adversary interacting in the real protocol (where no trusted third party exists) can do no more harm than if it was involved in the above-described ideal computation.

Execution in the ideal model. As we have mentioned, some malicious behavior cannot be prevented (for example, early aborting). This behavior is therefore incorporated into the ideal model. An ideal execution proceeds as follows:

Inputs: Each party obtains an input, denoted w ($w = x$ for P_i, and $w = y$ for P_j).

Send inputs to trusted party: An honest party always sends w to the trusted party. A malicious party may, depending on w, either abort or send some $w' \in \{0,1\}^{|w|}$ to the trusted party.

Trusted party answers first party: In case it has obtained an input pair (x, y), the trusted party first replies to the first party with $F_1(x, y)$. Otherwise (i.e., in case it receives only one valid input), the trusted party replies to both parties with a special symbol $\perp$.

Trusted party answers second party: In case the first party is malicious it may, depending on its input and the trusted party's answer, decide to stop the trusted party by sending it $\perp$ after receiving its output. In this case the trusted party sends $\perp$ to the second party. Otherwise (i.e., if not stopped), the trusted party sends $F_2(x, y)$ to the second party.

Outputs: An honest party always outputs the message it has obtained from the trusted party. A malicious party may output an arbitrary (probabilistic polynomial-time computable) function of its initial input and the message obtained from the trusted party.

Let $\mathcal{F} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ be a functionality where $\mathcal{F} = (F_1, F_2)$ and let $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ be a pair of non-uniform probabilistic expected polynomial-time machines (representing parties in the ideal model). Such a pair is *admissible* if for at least one

$i \in \{1,2\}$ we have that $\mathcal{S}_i$ is honest (i.e., follows the honest party instructions in the above-described ideal execution). Then, the *joint execution of F under $\mathcal{S}$ in the ideal model* (on input pair (x,y) and security parameter κ), denoted $\text{IDEAL}_{\mathcal{F},\mathcal{S}}(\kappa,x,y)$ is defined as the output pair of $\mathcal{S}_1$ and $\mathcal{S}_2$ from the above ideal execution.

Execution in the real model. We next consider the real model in which a real (two-party) protocol is executed (and there exists no trusted third party). In this case, a malicious party may follow an arbitrary feasible strategy; that is, any strategy implementable by non-uniform probabilistic polynomial-time machines. In particular, the malicious party may abort the execution at any point in time (and when this happens prematurely, the other party is left with no output). Let $\mathcal{F}$ be as above and let Π be a two-party protocol for computing $\mathcal{F}$. Furthermore, let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a pair of non-uniform probabilistic polynomial-time machines (representing parties in the real model). Such a pair is *admissible* if for at least one $i \in \{1,2\}$ we have that $\mathcal{A}_i$ is honest (i.e., follows the strategy specified by Π). Then, the *joint execution of Π under $\mathcal{A}$ in the real model*, denoted $\text{REAL}_{\Pi,\mathcal{A}}(\kappa,x,y)$, is defined as the output pair of $\mathcal{A}_1$ and $\mathcal{A}_2$ resulting from the protocol interaction.

Security as emulation of a real execution in the ideal model. Having defined the ideal and real models, we can now define security of protocols. Loosely speaking, the definition asserts that a secure two-party protocol (in the real model) emulates the ideal model (in which a trusted party exists). This is formulated by saying that admissible pairs in the ideal model are able to simulate admissible pairs in an execution of a secure real-model protocol.

Definition 7 (secure two-party computation)**.** Let $\mathcal{F}$ and Π be as above. Protocol Π is said to securely compute $\mathcal{F}$ (in the malicious model) if for every pair of admissible non-uniform probabilistic polynomial-time machines $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for the real model, there exists a pair of admissible non-uniform probabilistic expected polynomial-time machines $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ for the ideal model, such that:

$$\{\text{IDEAL}_{\mathcal{F},\mathcal{S}}(\kappa,x,y)\}_{\kappa\in\mathbb{N},x,y \text{ s.t. } |x|=|y|} \overset{\text{c}}{\approx} \{\text{REAL}_{\Pi,\mathcal{A}}(\kappa,x,y)\}_{\kappa\in\mathbb{N},x,y \text{ s.t. } |x|=|y|}$$

We note that the above definition assumes that the parties know the input lengths (this can be seen from the requirement that $|x| = |y|$). Some restriction on the input lengths is unavoidable, see [Gol04, Section 7.1] for discussion. We also note that we allow the ideal adversary/simulator to run in expected (rather than strict) polynomial-time. This is essential for constant-round protocols [BL04].

Protocol Π_{MPC}

Private Inputs: For $i \in [N]$, party P_i has input x_i.

Round 1: For $i \in [N]$ each party P_i proceeds as follows:

1. Choose randomness $r_i = (r_i^{gen}, r_i^{enc})$ for the MFHE protocol.
2. Choose an unrelated κ-bit randomness value R_i, and set $\hat{R}_i = OWF(R_i)$.
3. For every j, engage in a two-round commitment protocol with P_j for the values (x_i, r_i, R_i), using an instance of aCom with tag id_i. Note that the first message in this protocol is sent by P_j (so P_i sends the first message to all the P_j's for their respective commitments). Denote the message send from P_i to P_j by $\mathsf{acom}_1^{i,j}$.
4. For every j, prepare the first message $\mathsf{p}_1^{i,j}$ of Π_{WIPOK} (acting as the Prover) for the $\mathcal{NP}$-Language $\mathcal{L}_{P_i}^{\text{WIPOK}} = \mathcal{L}_{i,j,1} \vee \mathcal{L}_{i,j,2}$ for $j \in [N] \setminus \{i\}$ and the first message $\mathsf{fs}_1^{i,j}$ of Π_{FS} (acting as the Verifier) for $\mathcal{L}_{P_i}^{\text{FS}} = (\mathcal{L}_{i,j,1} \wedge \mathcal{L}_{i,j,3})$ where the $\mathcal{NP}$-Languages $\mathcal{L}_{i,j,1}$,$\mathcal{L}_{i,j,2}$,$\mathcal{L}_{i,j,3}$ are defined in Figure 4.
5. Generate the message α_i of the 1-round Π_{GenSetup} protocol.
6. For all j broadcast the message $m_1^{i,j} := \left(\hat{R}_i, \mathsf{acom}_1^{i,j}, \mathsf{p}_1^{i,j}, \mathsf{fs}_1^{i,j}, \alpha_i\right)$ to party P_j.

Round 2: For $i \in [N]$ each party P_i proceeds as follows:

1. Generate the second commitment messages $\mathsf{acom}_2^{j,i}$ for $\mathsf{aCom}_{\mathsf{id}_i}(x_i, r_i, R_i)$, the second message $\mathsf{p}_2^{j,i}$ of the Π_{WIPOK} proof system, and the second message $\mathsf{fs}_2^{j,i}$ of the Π_{FS} proof system.
2. For every j, engage in a two-round commitment protocol with P_j for the value 0, using an instance of bCom with tag id_i. As before, P_i sends the first message to all the P_j's for their respective commitments, and we denote the message send from P_i to P_j by $\mathsf{bcom}_1^{i,j}$.
3. For all j broadcast the messages $m_2^{i,j} := (\mathsf{acom}_2^{j,i}, \mathsf{bcom}_1^{i,j}, \mathsf{p}_2^{j,i}, \mathsf{fs}_2^{j,i}, \mathsf{pk}_i)$.

Round 3: For $i \in [N]$ each party P_i proceeds as follows:

1. Generate the second messages $\mathsf{bcom}_2^{j,i}$ corresponding to all $\mathsf{bCom}_{\mathsf{id}_i}(0)$, the final message $\mathsf{p}_3^{i,j}$ of the Π_{WIPOK} protocol, and the third message $\mathsf{fs}_3^{i,j}$ of Π_{FS}.
2. Compute the public matrix A from all the α_i's sent in the first round. Use randomness r_i^{gen}, r_i^{enc} to generate a key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ relative to A, and an encryption of the private input x_i, $c_i = \mathsf{MFHE.Encrypt}(pk_i, x_i)$.
3. For j broadcast the message $m_3^{i,j} := (\mathsf{bcom}_2^{j,i}, c_i, \mathsf{p}_3^{i,j}, \mathsf{fs}_3^{i,j})$.

Round 4: If any $\mathsf{p}^{j,i}$ does not pass verification then abort. Otherwise each party P_i:

1. Compute the evaluated ciphertext $\hat{c} := \text{MFHE.Eval}(\mathsf{params}; F; (c_1, \ldots, c_N))$, and the decryption shares $ev_i \leftarrow \mathsf{MFHE.PartDec}(\hat{c}, (\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i, \mathsf{sk}_i)$.
2. Prepare the final message $\mathsf{fs}_4^{j,i}$ of Π_{FS} protocol.
3. For all j, broadcast the message $m_4^{i,j} := (ev_i, \mathsf{fs}_4^{j,i})$.

Output phase: If any $\mathsf{fs}^{j,i}$ does not pass verification then abort. Else run the combining algorithm on the decryption shares, and output $y \leftarrow \mathsf{MFHE.FinDec}(ev_1, \ldots, ev_N, \hat{c})$.

Figure 1: Protocol Π_{MPC} with respect to party P_i.

$\mathcal{NP}$-Language $\mathcal{L}_{P_i}^{\text{WIPOK}}$ and $\mathcal{L}_{P_i}^{\text{FS}}$ for Π_{FS} and Π_{WIPOK} proof systems:

Fix the identities id_i, and then for all i, j define:

$$\mathcal{L}_{i,j,1} = \left\{ \left(\begin{array}{c} \hat{R}_i, \hat{R}_j, A, \mathsf{acom}_1^{j,i}, \mathsf{bcom}_1^{j,i} \\ \mathsf{pk}_i, c_i, \hat{c}, \mathsf{acom}_2^{j,i}, \mathsf{bcom}_2^{j,i} \end{array} \right) \middle| \begin{array}{l} \exists\ (x_i, r_i^{gen}, r_i^{enc}, \mathsf{sk}_i, R_i, \omega_i) : \\ \mathsf{acom}_2^{j,i} = \mathsf{aCom}_{\mathsf{id}_i}(x_i, r_i^{gen}, r_i^{enc}, R_i; \mathsf{acom}_1^{j,i}; \omega_i) \\ \wedge\ \hat{R}_i = OWF(R_i) \\ \wedge\ (\mathsf{sk}_i, \mathsf{pk}_i) = \mathsf{MFHE.Keygen}(A; r_i^{gen}) \\ \wedge\ c_i = \mathsf{MFHE.Encrypt}(\mathsf{pk}_i, x_i; r_i^{enc}) \end{array} \right\}$$

$$\mathcal{L}_{i,j,2} = \left\{ \left(\begin{array}{c} \hat{R}_i, \hat{R}_j, A, \mathsf{acom}_1^{j,i}, \mathsf{bcom}_1^{j,i} \\ \mathsf{pk}_i, c_i, \hat{c}, \mathsf{acom}_2^{j,i}, \mathsf{bcom}_2^{j,i} \end{array} \right) \middle| \begin{array}{l} \exists\ (R', \zeta) :\ \hat{R}_j = OWF(R') \\ \wedge\ \mathsf{bcom}_2^{j,i} = \mathsf{bCom}_{\mathsf{id}_i}(R'; \mathsf{bcom}_2^{j,i}; \zeta_i) \end{array} \right\}$$

$$\mathcal{L}_{i,j,3} = \left\{ \left(\begin{array}{c} \hat{R}_i, \hat{R}_j, A, \mathsf{acom}_1^{j,i}, \mathsf{bcom}_1^{j,i} \\ \mathsf{pk}_i, c_i, \hat{c}, \mathsf{acom}_2^{j,i}, \mathsf{bcom}_2^{j,i} \end{array} \right) \middle| \begin{array}{l} \exists\ (x_i, r_i^{gen}, r_i^{enc}, \mathsf{sk}_i, R_i, \omega_i) : \\ \mathsf{acom}_2^{j,i} = \mathsf{aCom}_{\mathsf{id}_i}(x_i, r_i^{gen}, r_i^{enc}, R_i; \mathsf{acom}_1^{j,i}; \omega_i) \\ \wedge\ (\mathsf{sk}_i, \mathsf{pk}_i) = \mathsf{MFHE.Keygen}(A; r_i^{gen}) \\ \wedge\ ev_i = \mathsf{MFHE.PartDec}(\hat{c}, i, \mathsf{sk}_i) \end{array} \right\}$$

We define $\mathcal{L}_{P_i}^{\text{WIPOK}} = \{\mathcal{L}_{i,j,1} \vee \mathcal{L}_{i,j,2}\}_j$ and $\mathcal{L}_{P_i}^{\text{FS}} = \{\mathcal{L}_{i,j,3}\}_j$.

Figure 2: $\mathcal{NP}$-Language $\mathcal{L}_{i,j,1}$,$\mathcal{L}_{i,j,2}$,$\mathcal{L}_{i,j,3}$ for Π_{FS} and Π_{WIPOK} proof systems.

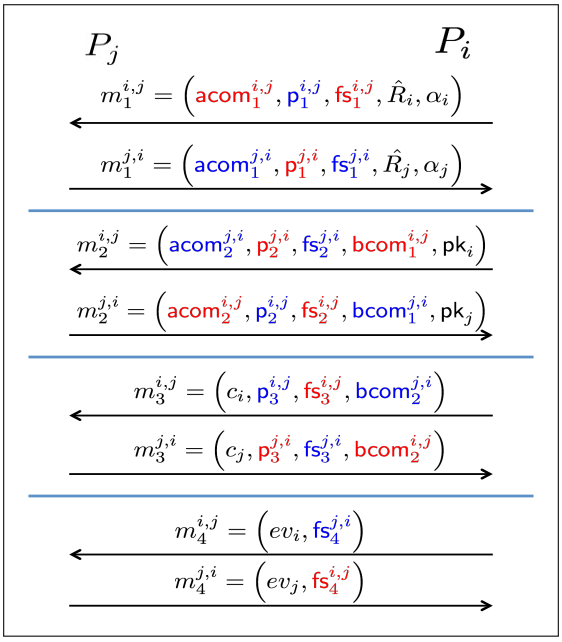

Figure 3: Messages exchanged between party P_i and P_j in Π_{MPC}. $(\mathsf{acom}_1, \mathsf{acom}_2)$ and $(\mathsf{bcom}_1, \mathsf{bcom}_2)$ are commitments, $(\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3)$ belong to the 3-round Π_{WIPOK}, $(\mathsf{fs}_1, \mathsf{fs}_2, \mathsf{fs}_3, \mathsf{fs}_4)$ belong to the 4-round Π_{FS}, and $(\alpha, \mathsf{pk}, c, ev)$ denote the MFHE messages. Blue messages are sub-protocols where party P_i is the prover/committer and party P_j is the verifier/receiver, red messages are the opposite.