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 in the plain model for any functionality, secure against a malicious adversary. Our protocol relies on the sub-exponential hardness of the Learning with Errors (LWE) problem with slightly super-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 and other strong assumptions.

To do this, we construct an LWE based multi-key FHE scheme with a very simple one-round *distributed setup procedure* (vs. the trusted setup required in previous LWE based constructions). This lets us construct a 3-round semi-malicious protocol without setup using the approach of Mukherjee and Wichs (EUROCRYPT '16). Finally, subexponential hardness and adaptive commitments are 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) allows mutually suspicious parties to evaluate a function on their joint private inputs without revealing these inputs to each other. One fruitful line of investigation is concerned with the *round complexity* of these protocols. More specifically, we consider a model where at each round, each party is allowed to broadcast a message to everyone else. We allow the adversary to be malicious and corrupt any fraction of the parties.

If a Common Random String (CRS) is allowed, then two rounds are necessary and sufficient for secure multi-party protocols under plausible cryptographic assumptions [GGHR14, MW16].

Without relying on trusted setup, it was still known that constant round protocols are possible [BMR90], but the exact round complexity remained open. A lower bound in the simultaneous message model was established in the recent work of Garg et al. [GMPP16], who proved that four rounds are necessary. They also showed how to perform multi-party coin flipping in four rounds (under strong assumptions), whic can then be used to generate a CRS and execute the aforementioned protocols in the CRS model. That technique implied a five-round protocol based on 3-round 3-robust non-malleable commitments[1] and indistinguishability obfuscation, and a 6 round protocol based on 3-round 3-robust non-malleable commitments and LWE.

For the important special case of only two parties, it is known that two message protocols with sequential rounds, i.e. each party talks in turn, are necessary and sufficient in the CRS model [JS07, HK07] and five message protocols are necessary and sufficient without setup [KO04, ORS15].

Our Results. Our work addresses the following fundamental question:

Can we obtain round-optimal multi-party computation protocols without setup?

We answer this 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. Informally, 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.

To establish this result, we depart from the coin flipping approach of [GMPP16] and instead rely on a new generalized notion of multi-key fully homomorphic encryp-

[1] Such commitments can be instantiated from adaptive PRGs [PPV08] (adaptive commitments) or one-way permutations secure w.r.t. sub-exponential time adversaries [COSV16].

tion [LTV12] which we show how to construct based on the hardness of LWE. In a nutshell, whereas prior LWE based constructions required trusted setup (essentially a CRS) [CM15, MW16, BP16, PS16], we show that the setup procedure can be distributed. Each party only needs to broadcast a random string[2], and generate its public key based on the collection of strings by all other parties. We show that the resulting scheme is secure even when some of the broadcast strings are adversarial (and even when the adversary is *rushing*). Similarly to Mukherjee and Wichs [MW16], we can transform our multi-key FHE into an MPC protocol in the *semi-malicious* model (where the adversary is only allowed to corrupt parties in a way that is consistent with *some* random tape). Our protocol requires 3 rounds without setup (vs. 2 rounds in the CRS model), and only requires *polynomial* hardness of LWE with slightly super polynomial noise ratio.

To get security in the malicious adversary model, we rely on other strong assumptions. In particular, we use a two-round *adaptively secure* commitment scheme (e.g., as constructed by Pandey, Pass and, Vaikuntanathan [PPV08], using Naor's protocol [Nao91] with adaptive PRGs). Moreover, we need a sub-exponential version of the LWE assumption.

Concurrent work. In a concurrent and independent work, Ananth, Choudhuri, and Jain construct a maliciously-secure 4-round MPC protocol based on one-way permutations and sub-exponential hardness of DDH [ACJ17]. Their approach is very different from ours, they construct and use a "robust semi-honest" MPC protocol from DDH, while our main building block is an LWE-based 3-round protocol against semi-malicious adversaries.

2 Overview of our Protocol

Our starting point is the multi-key FHE approach to MPC, first introduced by [LTV12]. As explained above, it was shown in [MW16] that the Clear-McGoldrick scheme [CM15] implies a 2 round protocol in the semi-malicious setting in the CRS model under LWE, and using NIZK it is possible to also achieve fully malicious security. Constructing a multi-key FHE without setup and with the necessary properties for compiling it into an MPC protocol is still an open problem, but we show that the trusted setup can be replaced by a distributed process which only adds a single round to the MPC protocol. While our final solution is quite simple, it relies on a number of insights as to the construction and use of multi-key FHE.

1. While the schemes in [GSW13, CM15, MW16] rely on *primal* Regev-style LWE-based encryption as a basis for their FHE scheme, it is also possible to instantiate them based on the *dual*-Regev scheme [GPV08] (with asymptotically similar performance). However, the same form of CRS is required for this instantiation as well, so at first glance it does not seem to offer any advantages.

[2] Essentially its public matrix for a dual-Regev encryption scheme [Reg09, GPV08].

2. The multi-key FHE schemes of [CM15, MW16] are presented as requiring trusted setup, but a deeper look reveals that this trusted setup is only needed to ensure a single property, related to the functionality: In LWE based encryption (Regev or dual-Regev) the public key contains a matrix A and a vector $t \cdot A$ (possibly with some additional noise, depending on the flavor of the scheme) where t is the secret key. In order to allow multi-key homomorphism between parties that each have their own A_i, t_i, it is only required that the values $b_{i,j} = t_i A_j$ for all i, j, are known to all participating parties (upto small noise). The use of CRS in previous works comes in setting all A_i to be the same matrix A, which is taken from the CRS, and thus the public key $b_i \approx t_i A_i = t_i A$ is the only required information.

3. Lastly, we notice that dual-Regev with the appropriate parameters is *leakage resilient* [DGK+10]. This means that so long as a party honestly generates its t_i and A_i, it can expose *arbitrary* (length-bounded) information on t_i without compromising the security of its own ciphertexts. Combining this with the above, we can afford to expose all $b_{i,j} = t_i A_j$ without creating a security concern (for appropriate setting of the parameters).

Putting the above observations together, we present a multi-key FHE scheme with a *distributed setup*, in which each party generates a piece of the common setup, namely the matrix A_i. After this step, each party can generate a public key pk_i containing all vectors $b_{i,j}$, the respective secret key is the vector t_i. Given all pk_i and the matrices A_i, it is possible to perform multi-key homomorphism in a very similar manner to [CM15, MW16]. The 3-round semi-malicious protocol is therefore as follows.

Round 1: Distributed Setup. Every player P_i broadcasts a random matrix A_i of the appropriate dimension.

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: Partial 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 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 shown 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 matrices A_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 correct encryption to complete before the parties send their decryption shares (and of course the proof of correct decryption must be completed before the output can be produced). Hence, if we have a k-round proof of correct encryption (and a $(k+1)$-round proof of correct decryption) then we get a $(k+1)$-round protocol overall. Much of the technical difficulties to achieve malicious security in the current work are related to using 3-round proofs of correct encryptions, resulting in a 4-round protocol.

2.1 The Maliciously-Secure Protocol

Our maliciously-secure protocol builds on the above 3-round semi-malicious protocol, and in addition it uses a two-round adaptive commitment protocol $\mathsf{aCom} = (\mathsf{acom}_1, \mathsf{acom}_2)$, a three-round proof of correct encryption $\Pi_{\text{WIPOK}} = (\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3)$, and a four-round proof of correct decryption $\Pi_{\text{FS}} = (\mathsf{fs}_1, \mathsf{fs}_2, \mathsf{fs}_3, \mathsf{fs}_4)$. (The names Π_{WIPOK} and Π_{FS} are meant to hint on the implementation of these proofs, see more discussion in the next subsection.)

Round 1: Distributed Setup, commitment & proof. Every party i broadcasts its setup matrix A_i. It also broadcasts the first message acom_1 of the adaptive commitment for its randomness and input, the first message p_1 of the proof of correct encryption, and the first message fs_1 of the proof of correct decryption (both proofs with respect to the committed values).

Round 2: Continued commitment & proofs. Each party broadcasts $\mathsf{acom}_2, \mathsf{p}_2, \mathsf{fs}_2$.

Round 3: Encryption & proofs. The parties collect all the first round matrices A_i and run the key-generation and encryption procedures of the multi-key FHE. Then, each party broadcasts its public key and encrypted input. In the same round, each party also broadcasts messages $\mathsf{p}_3, \mathsf{fs}_3$.

Round 4: Verification & decryption. Each party runs the verifier algorithm for the Π_{WIPOK} proof of correct encryption, verifying all the instances. If all of them passed then it evaluates the function on the encrypted inputs, then uses 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 correct 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 correct 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.

As explained in the next section, subtle technicalities arise in the security proof that lead to an extended version of the above protocol description.

2.2 A Tale of Malleability and Extraction

To prove security of our 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 correct 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 two-round adaptive commitments, 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*. Such schemes where constructed by Pandey et al. [PPV08], using Naor's scheme [Nao91] with adaptive PRGs.

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. Looking ahead, straight-line extraction is used in some of our hybrids to fake the (WIPOK) zero knowledge proofs.

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.[3]

[3] 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.

Delayed-input proofs. In the three-round proofs for correct encryption and in the four-round proofs for correct 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 and squeeze rounds in order to achieve a four round protocol.

Fake proofs via Feige-Shamir. The simulator needs to fake the four-round proof of correct decryption on behalf of the correct 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 as follows: Each party chooses a random string R, encloses $\hat{R} = OWF(R)$ with its first-flow message, encloses R inside the commitment aCom (together with its input and randomness) and adds the statement $\hat{R} = OWF(R)$ to the list of things that it proves in the 3-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 bCom, and use it as a trapdoor to fake the proof in the POK protocol.

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.

Part I
3-Round Semi-Malicious Protocols

3 LWE-Based Multi-Key FHE with Distributed Setup

Notations. Throughout the text 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 often use $[n]$ to denote the set $\{1, ..., n\}$.

$d \leftarrow \mathcal{D}$ denotes the process of sampling d from the distribution $\mathcal{D}$ or, if $\mathcal{D}$ is a set, a uniform choice from it. For two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, we use $\mathcal{D}_1 \approx_{\mathrm{s}} \mathcal{D}_2$ to denote that they are statistically close, $\mathcal{D}_1 \approx_{\mathrm{c}} \mathcal{D}_2$ denotes computational indistinguishability, and $\mathcal{D}_1 \equiv \mathcal{D}_2$ denotes identical distributions.

3.1 Definitions

An encryption scheme is *multi-key homomorphic* if it can evaluate circuits on ciphertexts that are encrypted under different keys. Decrypting an evaluated ciphertext requires the secret keys of all the ciphertexts that were included in the computation. In more detail, a multi-key homomorphic encryption scheme (with trusted setup) consists of five procedures, $\mathsf{MFHE} = (\mathsf{MFHE.Setup}, \mathsf{MFHE.Keygen}, \mathsf{MFHE.Encrypt}, \mathsf{MFHE.Decrypt}, \mathsf{MFHE.Eval})$:

- **Setup** $\mathsf{params} \leftarrow \mathsf{MFHE.Setup}(1^\kappa)$**:** 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** $c \leftarrow \mathsf{MFHE.Encrypt}(\mathsf{pk}, x)$**:** On input pk and a plaintext message $x \in \{0,1\}^*$ output a "fresh ciphertext" c. (We assume for convenience that the ciphertext includes also the respective public key.)
- **Evaluation** $\hat{c} := \mathsf{MFHE.Eval}(\mathsf{params}; \mathcal{C}; (c_1, \ldots, c_\ell))$**:** 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 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 $\mathcal{C}$, we set $\mathsf{params} \leftarrow \mathsf{MFHE.Setup}(1^\kappa)$ 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

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

except with negligible probability (in κ) taken over the randomness of all these algorithms.[4]

Local decryption and simulated shares. A special property that we need of the multi-key FHE schemes from [CM15, MW16], 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 that takes these partial shares and outputs the plaintext, $x \leftarrow \mathsf{MFHE.FinDec}(ev_1, \ldots, ev_N, \hat{c})$.

Another property of these schemes 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 want the simulated shares to be statistically close to the shares produced by the local partial decryption procedures using the keys $\{sk_i\}_{i \in I}$, even conditioned on 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. As in [MW16], in our case too we only achieve simulatability of the basic scheme when all parties but one are corrupted (i.e., when the set I is a singleton).

3.1.1 Distributed Setup

The variant that we need for our protocol does not require the setup procedure to be run by a trusted entity, but rather it is run in a distributed manner by all parties in the protocol. In our definition we allow the setup to depend on the maximum number of users N. This restriction does not pose a problem for our application.

- **Distributed Setup** $\mathsf{params}_i \leftarrow \mathsf{MFHE.DistSetup}(1^\kappa, 1^N, i)$**:** On input the security parameter κ and number of users N, outputs the system parameters for the i-th player params_i.

The remaining functions have the same functionality as above, where $\mathsf{params} = \{\mathsf{params}_i\}_{i \in [N]}$, the key generation takes i as an additional parameter in order to specify which entry in params it refers to.

[4] We often 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.

3.1.2 Semantic security and simulatability

Semantic security for mult-ikey FHE is defined as the usual notion of semantic security. For the distributed setup variant, we require that semantic security for the i-th party holds even when all $\{\mathsf{params}_j\}_{j\in[N]\setminus\{i\}}$ are generated adversarially and possibly depending on params_i.

Namely, we consider a *rushing adversary* that chooses N and $i \in [N]$, then it sees params_i and produces params_j for all $j \in [N] \setminus \{i\}$. After this setup, the adversary is engaged in the usual semantic-security game, where it is given the public key, chooses two messages and is given the encryption of one of them, and it needs to guess which one was encrypted.

Simulatability is defined as before, but now the evaluated ciphertext is produced by the honest party interacting with the same rushing adversary (and statistical closeness holds even conditioned on everything that the adversary sees).

3.2 A "Dual" LWE-Based Multi-Key FHE with Distributed Setup

For our protocol we use an adaptation of the "dual" of the 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 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 distributions χ, χ' with χ' producing much larger errors than χ (but still much smaller than q). Specifically, consider the distribution

$$\chi'' = \{a \leftarrow \{0,1\}^m, b \leftarrow \chi^m, c \leftarrow \chi',\ \text{output } c - \langle a, b\rangle\}. \tag{1}$$

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' are discrete Gaussian distributions around zero with parameters p, p', respectively, such that p'/p is super-polynomial (in n).

- **Distributed Setup** $\mathsf{params}_i \leftarrow \mathsf{MFHE.DistSetup}(1^\kappa, 1^N, i)$**:** Set the parameters $q = \mathrm{poly}(N)\mathrm{superpoly}(n)$ (as needed for FHE correctness), $m > (Nn+1)\log q + 2\kappa$, and $w = m \log q$. Sample and output a random matrix $A_i \in \mathbb{Z}_q^{(m-1)\times n}$.

- **Key Generation** $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{MFHE.Keygen}(\mathsf{params}, i)$**:** Recall that $\mathsf{params} = \{\mathsf{params}_i\}_{i\in[N]} = \{A_i\}_{i\in[N]}$. The public key of party i is a sequence of vectors $\mathsf{pk}_i = \{b_{i,j}\}_{j\in[N]}$ to be formally defined below. The corresponding secret key is a *low-norm vector* $t_i \in \mathbb{Z}_q^m$.

We will define $b_{i,j}$, t_i such that for $B_{i,j} = \begin{pmatrix} A_j \\ -b_{i,j} \end{pmatrix}$ it holds that $t_i B_{i,j} = b_{i,i} - b_{i,j} \pmod q$ for all j.

In more detail, sample a random binary vector $s_i \leftarrow \{0,1\}^{m-1}$, we set $b_{i,j} = s_i A_j \bmod q$. Denoting $t_i = (s_i, 1)$, we indeed have $t_i B_{i,j} = b_{i,i} - b_{i,j} \pmod q$.

- **Encryption** $c \leftarrow \mathsf{MFHE.Encrypt}(\mathsf{pk}_i, \mu)$**:** To encrypt a bit μ under the public key pk_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

$$C := B_{i,i} R + E + \mu G \mod q, \tag{2}$$

  where G is a fixed m-by-w "gadget matrix" (whose structure is not important for us here , cf. [MP12]). Furthermore, as in [CM15, MW16], encrypt all bits of R in a similar manner. 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** $\mu := \mathsf{MFHE.Decrypt}((\mathsf{sk}_1, \ldots, \mathsf{sk}_N), \hat{c})$**:** The invariant satisfied by ciphertexts in this scheme, similarly to GSW, is that an encryption of a bit μ relative to secret key t is a matrix C that satisfies

$$tC = \mu \cdot tG + e \pmod q \tag{3}$$

  for a low-norm error vector e, where G is the same "gadget matrix". The vector t is the concatenation of all $\mathsf{sk}_i = t_i$ for all parties i participating in the evaluation.

  This invariant holds for freshly encrypted ciphertexts since $t_i B_{i,i} = 0 \pmod q$, and so $t_i(B_{i,i} R + E + \mu G) = \mu \cdot t_i G + t_i E \pmod q$, where $e = t_i E$ has low norm (as both t_i and E have low norm).

  To decrypt, the secret-key holders compute $u = t \cdot C \bmod q$, outputting 1 if the result is closer to tG or 0 if the result is closer to 0.

- **Evaluation** $\hat{c} := \mathsf{MFHE.Eval}(\mathsf{params}; \mathcal{C}; (c_1, \ldots, c_\ell))$**:** 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.

3.2.1 Security

Security with distributed setup follows from LWE so long as $(m-1) > (Nn+1)\log q + 2\kappa$. The basis for security is the following lemma, which is essentially the same argument from [DGK+10] showing that dual Regev is leakage resilient for bounded leakage.

Lemma 1. *Let $A_i \in \mathbb{Z}_q^{(m-1)\times n}$ be uniform, and let A_j for all $j \neq i$ be chosen by a rushing adversary after seeing A_i. Let $s_i \leftarrow \{0,1\}^{m-1}$ and $b_{i,j} = s_i A_j$. Let $r \in \mathbb{Z}_q^n$ be uniform, $e \leftarrow \chi^{m-1}$, $e' \leftarrow \chi'$. Then, under the LWE assumption, the vector $c = A_i r + e$ and number $c' = b_{i,i} r + e'$ are (jointly) pseudorandom, even given the $b_{i,j}$'s for all $j \in [N]$ and the view of the adversary that generated the A_j's.*

Proof: Consider the distribution of c, c' as in the lemma statement. We notice that $c' = b_{i,i} r + e' = s_i A_i r + e' = s_i c - s_i e + e'$. The proof proceeds by a sequence of hybrids. Our first hybrid changes the distribution of c' to $c' = s_i c + e'$. Noting that $c' - s_i c$ is drawn from χ'' before the change and from χ' after the change (cf. Eqn. (1)), we get that the statistical distance between the hybrids is negligible.

In the next hybrid, we use LWE to replace c with a uniform vector. Since c could have been sampled before s_i or any of the A_j with $j \neq i$, LWE implies indistinguishability with the previous hybrid.

Finally, we apply the leftover hash lemma, noting that all the $b_{i,j}$'s only leak at most $Nn \log q$ bits of information on s_i and therefore the average min-entropy of s_i is at least $(m-1) - Nn \log q > \log q + 2\kappa$. Using the leftover hash lemma with c as seed and s_i as source, we have that $(c, s_i c)$ are jointly statistically indistinguishable from uniform. This implies that (c, c') are jointly statistically indistinguishable from uniform, even given all $A_j, b_{i,j}$ for all $j \in [N]$. The lemma follows. □

Applying this lemma repeatedly for every column via a hybrid argument shows that the ciphertext components $c = A_i R + \hat{E}$ and $c' = b_{i,i} R + e'$ are also jointly pseudorandom, even given the view of the adversary, and semantic security of the scheme follows.

3.2.2 Multi-key Homomorphism and Simulatability

The other components of the multi-key FHE scheme from [CM15, MW16] work for our variant as well, simply because the encryption and decryption formulas are identical (except with slightly different parameter setting), namely Equations (2) and (3). Below we briefly sketch these components for the sake of self-containment.

The ciphertext-expansion procedure. The "gadget matrix" G used for these schemes has the property that there exists a low-norm vector u such that $Gu = (0, 0, \ldots, 0, 1)$. 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 $\langle t, v \rangle \approx \mu \pmod q$. Let A_1, A_2 be public parameters for two users with secret keys $t_1 = (s_1|1)$, $t_2 = (s_2|1)$, and recall that we denote $b_{i,j} = s_i A_j$ and $B_{i,i} = \begin{pmatrix} A_i \\ -s_i A_i \end{pmatrix} = \begin{pmatrix} A_i \\ -b_{i,i} \end{pmatrix}$.

Let $C = B_{1,1}R + E + \mu G$ be fresh encryption of μ w.r.t $B_{1,1}$, and suppose that we also have an encryption under t_1 of the matrix R. We note that given any vector δ, we can apply homomorphic operations to the encryption of R to get an encryption of the entries of the vector $\rho = \rho(\delta) = \delta R$. 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 $X = X(\delta)$ such that $t_1 X \approx \rho = \delta R \pmod q$.

We consider the matrix $C' = \begin{pmatrix} C & X \\ 0 & C \end{pmatrix}$, where $X = X(\delta)$ for a δ to be determined later. We claim that for an appropriate δ this is an encryption of the same plaintext μ under the concatenated secret key $t' = (t_1|t_2)$. To see this, notice that

$$t_2 C = (s_2|1)\left(\begin{pmatrix} A_1 \\ -s_1 A_1 \end{pmatrix} R + E + \mu G\right) \approx (b_{2,1} - b_{1,1})R + \mu t_2 G \pmod q,$$

and therefore setting $\delta = b_{1,1} - b_{2,1}$, which is a value that can be computed from $\mathsf{pk}_1, \mathsf{pk}_2$ we get

$$\begin{aligned} t'C' &= (t_1 C \mid t_1 X + t_2 C) \approx (\mu t_1 G \mid (b_{1,1} - b_{2,1})R + (b_{2,1} - b_{1,1})R + \mu t_2 G) \\ &= \mu(t_1 G \mid t_2 G) = \mu(t_1|t_2)\begin{pmatrix} G & \\ & G \end{pmatrix}, \end{aligned}$$

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.

Partial decryption and Simulatability. This aspect works exactly as in [MW16, Thm 5.6]. Let $\mathbf{v}$ be a fixed low-norm vector satisfying $G\mathbf{v} = (0, 0, \ldots, 0, \lceil q/2 \rceil) \pmod q$ (such a vector exists). Let C be an encryption of a bit μ relative to the concatenated secret key $t = (t_1|t_2|\ldots|t_N)$ (whose last entry is 1). Then on one hand C satisfies Eqn. (3) so we have

$$tC\mathbf{v} = \mu \underbrace{tG\mathbf{v}}_{=\lceil q/2 \rceil} + \underbrace{\langle e, \mathbf{v} \rangle}_{=\epsilon, |\epsilon| \ll q} \approx \mu \cdot \lceil q/2 \rceil \pmod q.$$

On the other hand, breaking C into N bands of m rows each (i.e, $C = (C_1^T|C_2^T|\ldots|C_N^T)^T$ with each $C_i \in \mathbb{Z}_q^{m \times mN}$), we have $tC\mathbf{v} = \sum_{i=1}^N t_i C_i \mathbf{v}$. Hence in principle we could set the partial decryption procedure as $ev_i = \mathsf{MFHE.PartDec}(C, t_i) := t_i C_i \mathbf{v} \bmod q$, and the combination procedure will just add all these ev_i's and output 0 if it is smaller than $q/4$ in magnitude and 1 otherwise.

To be able to simulate (when there are $N-1$ corruptions), we need the partial decryption to add its own noise, large enough to "drown" the noise in $tC\mathbf{v}$ (but small enough so decryption still works). Given the ciphertext C, $N-1$ keys t_j for all $j \in [N] \setminus \{i\}$, and the plaintext bit μ, the simulator will sample its own noise e and output the share $ev_i = \mu \cdot \lceil q/2 \rceil + e - \sum_j t_j C_j \mathbf{v} \bmod q$.

4 A Semi-Malicious Protocol without Setup

The semi-malicious adversary model [AJL+12] is a useful mid-point between the semi-honest and fully-malicious models. Somewhat similarly to a semi-honest adversary, a semi-malicious adversary is restricted to run the prescribed protocol, but differently than the semi-honest model it can choose the randomness that this protocol expects arbitrarily and adaptively (as opposed to just choosing it at random). Namely, at any point in the protocol, there must exists a choice of inputs and randomness that completely explain the messages sent by the adversary, but these inputs and randomness can be arbitrarily chosen by the adversary itself. A somewhat subtle point is that the adversary must always know the inputs and randomness that explain its actions (i.e., the model requires the adversary to explicitly output these before any messages that it sends).

We still assume a rushing adversary that can choose its messages after seeing the messages of the honest parties (subject to the constraint above). Similarly to the malicious model, an adversarial party can abort the computation at any point. Security is defined in the usual way, by requiring that a real-model execution is simulatable by an adversary/simulator in the ideal model, cf. Definition 7 in Section 5.3.

4.1 A Semi-Malicious Protocol from Multi-Key FHE with Distributed Setup

Our construction of 3-round semi-malicious protocol without setup is nearly identical to the Mukherjee-Wichs construction with a common reference string [MW16, Sec. 6], except that we use multi-Key FHE with distributed setup, instead of their multi-Key FHE with trusted setup. We briefly describe this construction here for the sake of self-containment.

- To compute an N-party function $\mathcal{F} : (\{0,1\}^*)^N \to \{0,1\}^*$ on input vector $\mathbf{w}$, the parties first run the setup round and broadcast their local parameters params_i.
- Setting $\mathsf{params} = (\mathsf{params}_1, \ldots, \mathsf{params}_N)$, each party runs the key generation to get $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{MFHE.Keygen}(\mathsf{params}, i)$ and then the encryption algorithm $c_i \leftarrow \mathsf{MFHE.Encrypt}(pk_i, w_i)$, and broadcasts (pk_i, c_i).
- Once the parties have all the public keys and ciphertexts, they each evaluate homomorphically the function $\mathcal{F}$ and all get the same evaluated ciphertext $\hat{c}$. Each party applies its partial decryption procedure to get $ev_i \leftarrow \mathsf{MFHE.PartDec}(\hat{c}, \mathsf{sk}_i)$ and broadcasts its decryption share ev_i to everyone.
- Finally, given all the shares ev_i, every party runs the combination procedure and outputs $\mu \leftarrow \mathsf{MFHE.FinDec}(ev_1, \ldots, ev_N, \hat{c})$.

Security. Security is argued exactly as in [MW16, Thm. 6.1]: First we use the simulatability property to replace the partial decryption by the honest parties by a simulated partial decryption (cf. [MW16, Lem. 6.2]), and once the keys of the honest parties are no longer needed we can appeal to the semantic security of the FHE scheme (cf. [MW16, Lem. 6.3]).

Exactly as in the Mukherjee-Wichs construction, here too the underlying multi-key scheme only satisfies simulatability when all but one of the parties are corrupted, and as a result also the protocol above is only secure against adversaries that corrupt all but one of the parties. Mukherjee and Wichs described in [MW16, Sec. 6.2] a transformation from a protocol secure against exactly $N-1$ corruptions to one which is secure against any number of corruptions. Their transformation is generic and can be applied also in our context, resulting in a semi-malicious-secure protocol.

Part II
4-Round Malicious Protocols

5 Tools and Definitions

We use tools of commitment and proofs to "compile" our semi-malicious protocol to a protocol secure in the malicious model. Below we define these tools and review the properties that we rely on.

5.1 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.

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

where $\mathsf{view}_{\mathsf{aCom}}^{R^*(\mathsf{tag}),\mathcal{B}_{\mathsf{tag}}}(m, z)$ denotes the random variable describing the output of $R^*(\mathsf{tag})$ after receiving a commitment to m relative to tag using aCom, 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 $\mathsf{aCom}_1 = \mathsf{aCom}_{\mathsf{tag}}(r)$ and $\mathsf{aCom}_2 = \mathsf{aCom}_{\mathsf{tag}}(m; \mathsf{aCom}_1; r')$ 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.

5.2 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 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.

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.

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 $\kappa \in \mathbb{N}$.

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

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 parametrized 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 [COSV16].

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.

5.3 Secure Computation

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 model". A protocol is secure if any adversary interacting in the real protocol can do no more harm than if it was involved in this "ideal" computation.

Execution in the ideal model. In the "ideal model" we have 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. Even this model is not completely "ideal", however, since some malicious behavior that cannot be prevented (such as early aborting) is permitted here too. An ideal execution proceeds as follows:

Inputs: Each party obtains an input, denoted w.

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 malicious parties: The trusted party is informed of the set of malicious parties M, and let us denote the complementing set of honest parties by H.

Once it received all the inputs, the trusted party first replies to the malicious parties with $F_M(\mathbf{w})$.

Trusted party answers second party: The malicious parties reply to the trusted party by either "proceed" or "abort". If they all reply "proceed" then the trusted party sends $F_H(\mathbf{w})$ to the honest parties. If any of them reply "abort" then the trusted party sends $\perp$ to the honest parties.

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

The random variable containing the joint outputs of the honest and malicious parties in this execution (including an identification of the set M of malicious parties) is denoted $\text{IDEAL}_{\mathcal{F},\mathcal{S}}(\kappa, \mathbf{w})$, where κ is the security parameter and $\mathbf{w}$ are the inputs.

Execution in the real model. In the real model ,where there is no trusted party, 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). The (static) adversary chooses the set M of malicious parties before it receives any inputs to the protocol, and it can be *rushing*, in that in every communication round it first sees the messages from the honest parties and only then chooses the messages on behalf of the malicious parties.

Let $\mathcal{F} : (\{0,1\}^*)^N \to (\{0,1\}^*)^N$ be an N-party function, let Π be an N-party protocol for computing $\mathcal{F}$, and let $\mathcal{A}$ be an adversary. The *joint execution of* Π *with adversary* $\mathcal{A}$ *in the real model*, denoted $\text{REAL}_{\Pi,\mathcal{A}}(\kappa, \mathbf{w})$ (with κ the security parameter and $\mathbf{w}$ the inputs), is defined as the output of the honest and malicious parties (and an identification of the set M of malicious parties), resulting from the protocol interaction.

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

$$\{\text{IDEAL}_{\mathcal{F},\mathcal{S}}(\kappa, \mathbf{w})\}_{\kappa\in\mathbb{N},\mathbf{w}\in(\{0,1\}^*)^N} \overset{\text{c}}{\approx} \{\text{REAL}_{\Pi,\mathcal{A}}(\kappa, \mathbf{w})\}_{\kappa\in\mathbb{N},\mathbf{w}\in(\{0,1\}^*)^N} .$$

Notations. For a sub-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 sub-protocol is sent by P_i to P_j. Likewise, subscripts (j, i) denote that the *first* message of the sub-protocol is sent by P_j to P_i.

6 A Malicious Protocol without Setup

Our 4-round protocol for the malicious case is obtained by "compiling" the 3-round semi-malicious protocol from Section 4, adding round-efficient proofs of correct behavior. The components of this protocol are:

- The 3-round semi-malicious protocol from Section 4, based on the "dual"-GSW-based multi-key FHE scheme with distributed setup. We denote this multi-key FHE scheme by $\mathsf{MFHE} = (\mathsf{MFHE.DistSetup}, \mathsf{MFHE.Keygen}, \mathsf{MFHE.Encrypt}, \mathsf{MFHE.Eval}, \mathsf{MFHE.PartDec}, \mathsf{MFHE.FinDec})$.

- 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 $\mathsf{bCom} = (\mathsf{bcom}_1, \mathsf{bcom}_2)$.
- A one-way function OWF.
- A three-round public coin witness-indistinguishable proof of knowledge with delayed input, $\Pi_{\text{WIPOK}} = (\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3)$, for the $\mathcal{NP}$-Language $\mathcal{L}_P^{\text{WIPOK}}$ from Figure 2. We often refer to this protocol as "proof of correct encryption", but what it really proves is that EITHER the encryption is consistent with the values committed in aCom, OR the value committed in bCom is a pre-image under OWF of values sent by the other parties.
- A four-round zero-knowledge argument of knowledge with delayed input, $\Pi_{\text{FS}} = (\mathsf{fs}_1, \mathsf{fs}_2, \mathsf{fs}_3, \mathsf{fs}_4)$, for the $\mathcal{NP}$-Language $\mathcal{L}_P^{\text{FS}}$ from Figure 2. We often refer to this protocol as "proof of correct decryption".

The parameters for the MFHE scheme, the OWF, and the two proof systems, are chosen polynomially larger than those for the commitment schemes. Hence (assuming sub-exponential security), all these constructions remain secure even against an adversary that can break $\mathsf{aCom}, \mathsf{bCom}$ by exhaustive search.

The protocol. Let $F : (\{0,1\}^*)^N \to \{0,1\}^*$ be a deterministic N-party function to be computed. Each party P_i holds input $x_i \in \{0,1\}^\kappa$ and identity id_i. [5] 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. 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.

[5] Known transformations yield also protocols for randomized functionalities without increasing the rounds, see [Gol04, Section 7.3].

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 scheme.
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 messages initiated in each sub-protocol by 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 $\mathcal{L}_{P_j}^{\text{FS}} = (\mathcal{L}_{j,i,1} \wedge \mathcal{L}_{j,i,3})$ where the $\mathcal{NP}$-Languages $\mathcal{L}_{i,j,1}$,$\mathcal{L}_{i,j,2}$,$\mathcal{L}_{i,j,3}$ are defined in Figure 6.
5. Run the distributed setup of MFHE to get $\mathsf{params}_i = \mathsf{MFHE.DistSetup}(1^\kappa, 1^N, i)$.
6. For all $j(\neq i)$ broadcast the message $m_1^{i,j} := \left(\mathsf{acom}_1^{i,j}, \mathsf{p}_1^{i,j}, \mathsf{fs}_1^{i,j}, \hat{R}_i, \mathsf{params}_i\right)$ 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 $\mathbf{0}$, 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 sent from P_i to P_j by $\mathsf{bcom}_1^{i,j}$.
3. For all j broadcast the messages $m_2^{i,j} := (\mathsf{acom}_2^{j,i}, \mathsf{p}_2^{j,i}, \mathsf{fs}_2^{j,i}, \mathsf{bcom}_1^{i,j})$.

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 $\mathsf{bCom}_{\mathsf{id}_i}(\mathbf{0})$, 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. Set $\mathsf{params} = \{\mathsf{params}_i\}_{i \in [N]}$. Use randomness r_i^{gen}, r_i^{enc} to generate a key pair for MFHE, $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{MFHE.Keygen}(\mathsf{params}, i)$, and an encryption of the private input $c_i = \mathsf{MFHE.Encrypt}(pk_i, x_i)$.
3. For all j broadcast the message $m_3^{i,j} := (\mathsf{pk}_i, c_i, \mathsf{p}_3^{i,j}, \mathsf{fs}_3^{i,j}, \mathsf{bcom}_2^{j,i})$.

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

1. Compute the evaluated ciphertext $\hat{c} := \mathsf{MFHE.Eval}(\mathsf{params}; F; (c_1, \ldots, c_N))$, 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 $\mathsf{fs}^{i,j}$ 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 where P_i acts as the prover:

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

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

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

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

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.

6.1 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 a 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. We also note that while we think of the protocol from Figure 1 as a "compilation" of the 3-round protocol from Section 4 using zero-knowledge proofs, it is not a generic compiler, as it relies on the specifics of our semi-malicious protocol. See more discussion in Section 6.2 below.

Below we prove security of Π_{MPC} by describing a simulator and proving that the simulated view is indistinguishable from the real one.

6.1.1 Description of the Simulator

Let $\mathcal{P} = \{P_1, \ldots, P_N\}$ be the set of parties, let $\mathcal{A}$ be a malicious, static adversary in the plain model, and let $\mathcal{P}^* \subseteq \mathcal{P}$ be the set of parties corrupted by $\mathcal{A}$. We construct a simulator

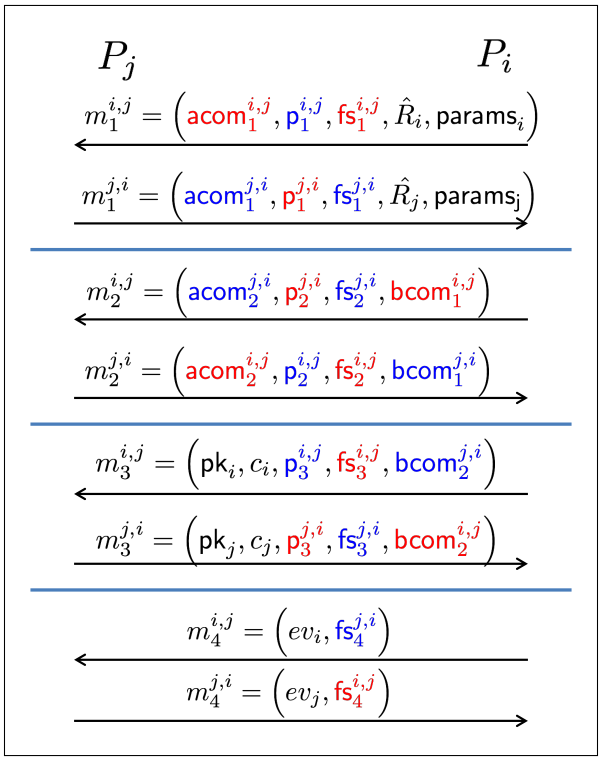

Figure 3: Messages exchanged between party P_i and P_j in Π_{MPC}. $(\mathsf{acom}_1, \mathsf{acom}_2)$ and $(\mathsf{bcom}_1, \mathsf{bcom}_2)$ are commitments, $(\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3)$ belong to the 3-round Π_{WIPOK}, $(\mathsf{fs}_1, \mathsf{fs}_2, \mathsf{fs}_3, \mathsf{fs}_4)$ belong to the 4-round Π_{FS}, and $(\mathsf{params}, \mathsf{pk}, c, ev)$ 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.

$\mathcal{S}$ (the ideal world adversary) with access to the ideal functionality $\mathcal{F}$, such that the ideal world experiment with $\mathcal{S}$ and $\mathcal{F}$ is indistinguishable from a real execution of Π_{MPC} with $\mathcal{A}$. The simulator $\mathcal{S}$ only generates messages on behalf of parties $\mathcal{P} \backslash \mathcal{P}^*$, as follows:

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 scheme 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 commit-

ment. 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} (with P_h as Prover) for the $\mathcal{NP}$-Language $\mathcal{L}_{P_h}^{\text{WIPOK}}$, and the first message $\mathsf{fs}_1^{h,j}$ of Π_{FS} (with P_h as Verifier) for $\mathcal{L}_{P_j}^{\text{FS}}$.

4. Run $\mathsf{params}_h \leftarrow \mathsf{MFHE.DistSetup}(1^\kappa, 1^N, h)$.

5. Send the message $m_1^{h,j} = \left(\mathsf{acom}_1^{h,j}, \mathsf{p}_1^{h,j}, \mathsf{fs}_1^{h,j}, \hat{R}_h, \mathsf{params}_h\right)$ to $\mathcal{A}$.

Round 1 Messages $\mathcal{A} \to \mathcal{S}$: Also in the first round the adversary $\mathcal{A}$ generates the messages $m_1^{j,h} = \left(\mathsf{acom}_1^{j,h}, \mathsf{p}_1^{j,h}, \mathsf{fs}_1^{j,h}, \hat{R}_j, \mathsf{params}_j\right)$ 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. Send the message $m_2^{h,j} = (\mathsf{acom}_2^{j,h}, \mathsf{p}_2^{j,h}, \mathsf{fs}_2^{j,h}, \mathsf{bcom}_1^{h,j})$ to $\mathcal{A}$.

Round 2 Messages $\mathcal{A} \to \mathcal{S}$: In the second round the adversary $\mathcal{A}$ generates the messages $m_2^{j,h} := (\mathsf{acom}_2^{h,j}, \mathsf{p}_2^{h,j}, \mathsf{fs}_2^{h,j}, \mathsf{bcom}_1^{j,h})$ 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. Set $\mathsf{params} = (\mathsf{params}_1, \ldots, \mathsf{params}_N)$ for the MFHE scheme and generate the keys $(\mathsf{pk}_h, \mathsf{sk}_h) = \mathsf{MFHE.Keygen}(\mathsf{params}, h; r_h^{gen})$. Generate an encryption of zero using randomness r_h^{enc}, $c_h = \mathsf{MFHE.Encrypt}(pk_h, \mathbf{0}; r_h^{enc})$.

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

Round 3 Messages $\mathcal{A} \to \mathcal{S}$: $\mathcal{S}$ receives $m_3^{j,h} = (\mathsf{pk}_j, c_j, \mathsf{p}_3^{j,h}, \mathsf{fs}_3^{j,h}, \mathsf{bcom}_2^{h,j})$ from $\mathcal{A}$, 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 $(\mathsf{p}_1^{j,h}, \mathsf{p}_2^{j,h}, \mathsf{p}_3^{j,h})$ completed in the first three rounds, using 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) [6]. $\mathcal{S}$ also uses the zero-knowledge simulator of Π_{FS} to obtain the "trapdoors" associated with that protocol. (Note that here we rely on the specific structure of Feige-Shamir proofs, where the zero-knowledge simulator extracts a "verifier secret" after the 3rd round, that makes it possible to simulate the last round.)

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 $y = F(\{x_j\}_{j\in[N]})$.

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 $\hat{c} := \mathsf{MFHE.Eval}(\mathsf{params}; F; (c_1, \ldots, c_N))$.

2. $\mathcal{S}$ reconstructs all the secret keys $\{\mathsf{sk}_j\}_{j\in\mathcal{P}^*}$ from the witnesses $\{r_j^{gen}\}_{j\in\mathcal{P}^*}$, 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, Sec. 6.2].[7])

3. Simulate the final message $\mathsf{fs}_4^{j,h}$ of Π_{FS} protocol using the extracted trapdoor.

   $\mathcal{S}$ 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 $m_4^{j,h} = (ev_j, \mathsf{fs}_4^{h,j})$ from $\mathcal{A}$.

This completes the description of the simulator.

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

[7] To use $\mathcal{S}^T$ from [MW16, Sec. 6.2] we need to evaluate the protocol on a different function $\mathcal{F}'$ rather than $\mathcal{F}$, we ignore this detail in the rest of the presentation here.

6.1.2 Proof of Indistinguishability

Overview. 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:

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

To prove indistinguishability, we consider a sequence of hybrid experiments. Let H_0 be the hybrid describing the real-world execution of the protocol, and we modify it in steps:

- 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 one of the honest parties). 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.
- H_5 Encrypt $\mathbf{0}$'s rather than the true inputs. Indistinguishability follows due to the semantic security of the encryption scheme.
- 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 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 aCom 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 aCom 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 equal to the number of rewinding operations. We now proceed to give more details.

$\mathbf{H_0}$: 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 6.1. $\mathrm{H}_0 \approx_{\mathrm{s}} \mathrm{H}_1$.

$\mathbf{H_2}$: 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).[8]

[8]The commitment bCom starts in the second round, but this is a two-round commitment so the com-

Lemma 6.2. $\mathrm{H}_1 \approx_c \mathrm{H}_2$.

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 $\mathrm{H}_1 \approx_c \mathrm{H}_2$ with probability ϵ then the reduction algorithm can distinguish commitments to 0 and R_j with probability ϵ/M. □

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 6.3. $\mathrm{H}_2 \approx_c \mathrm{H}_3$.

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. □

mitted value only affects the second message in the commitment, which happens in the third round of the larger protocol.

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 6.4. $\mathrm{H}_3 \approx_{\mathrm{s}} \mathrm{H}_4$.

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. □

$\mathbf{H_5}$: 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 6.5. $\mathrm{H}_4 \approx_{\mathrm{c}} \mathrm{H}_5$.

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 parties' 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 6.6. $\mathrm{H}_5 \approx_c \mathrm{H}_6$.

Proof: This proof is very similar to the the proof of $\mathrm{H}_1 \approx_c \mathrm{H}_2$, 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 $x_h = \mathbf{0}$. As in Lemma 6.3 indistinguishability follows by the WI property of Π_{WIPOK}.

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 6.2.

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 6.7. $\mathrm{H}_8 \approx_s \mathrm{H}_9$.

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. □

Observing that the hybrid H_9 is identical to the ideal-world game with the simulator completes the proof of security. □

6.2 Discussion and Open Problems

Compiling semi-malicious to malicious protocols. Our protocol and its proof can be viewed as starting from a 3-round semi-malicious protocol and "compiling" it into a 4-round malicious protocol using commitments and zero-knowledge proofs. However our construction *is not* a generic compiler of semi-malicious to malicious protocols, rather it relies on the specifics of our 3-round semi-malicious protocol from Section 4. At the very least, our construction needs the following two properties of the underlying semi-malicious protocol:

Public-coin 1st round. In our protocol we must send the second-round messages of the underlying protocol no later than the 3rd round of the compiled protocol. We thus have at most two rounds to prove that the first-round messages are valid, before we must send the second-round messages, severely limiting the type of proofs that we can use.

This is not a problem in our case, since the first round of the semi-malicious protocol is *public coin*, i.e., the parties just send to each other random bits. Hence, there is nothing to prove about them and the semi-malicious protocol can withstand any messages sent by the adversary.

Committing 2nd round. We also use the fact that the second round of the semi-malicious protocol is fully committing to the input, since our simulator extracts the inputs after these rounds.

We remark that in some sense every 3-round semi-malicious protocol with a public-coin first round and fully-committing second round can be thought of as a multi-key homomorphic encryption with distributed setup, by viewing the random coins send in the first round as the params, and the second-round messages as encryptions of the inputs.

Adaptive commitments. Although the intuitive property that we need from the commitment component of our protocol is non malleability, our actual proof relies heavily on the stronger notion of *adaptive security*, that lets us use straight-line extraction from the adversary's commitment. While it is plausible that our 3-round semi-malicious protocol can be "compiled" using only non-malleable commitments and avoid complexity leveraging,

we were not able to do it, this question remains open.[9]

Another open problem is to base two-round adaptive commitments on more standard assumptions. The only construction we have is the original one due to Pandey et al. from adaptive PRGs [PPV08]. It is plausible that the new two-round commitment scheme of Lin, Pass, and Soni [LPS17] from "Time-Lock Puzzles" can be made adaptively secure, though it was not claimed in [LPS17].

References

[ACJ17] Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. Cryptology ePrint Archive, Report 2017/402, 2017. http://eprint.iacr.org/2017/402.

[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.

[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.

[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.

[DGK+10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs.

[9] The concurrent work of [ACJ17] achieves a "compilation" of their robust semi-honest protocol to the malicious setting based on complexity leveraging and non-malleable commitments.

In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 361–381. Springer, Heidelberg, February 2010.

[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.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 74–94, 2014.

[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.

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

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. ACM, 2008.

[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.

[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.

[LPS17] Huijia Lin, Rafael Pass, and Pratik Soni. Two-round concurrent non-malleable commitment from time-lock puzzles. *IACR Cryptology ePrint Archive*, 2017:273, 2017.

[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.

[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.

[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.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

A The Need for Dual GSW

For the interested reader, we explain below why we need to use the "dual" rather than "primal" GSW scheme for our multi-key FHE. The main difference is that in the scheme from [CM15, MW16], the common matrix A has dimension $(n-1)$-by-m (with $m > n$), while in our scheme the dimensions are flipped and the matrix $A = (A_1|\ldots|A_n)$ is of dimension $(m-1)$-by-Nn with $m > Nn$. While it is possible that a secure one-round distributed setup procedure 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 in which each party P_i is choosing a random $n \times m'$ matrix A_i ($n < m'$), and then using the column-concatenation of all the A_i's, $A = (A_1|A_2|\ldots|A_N)$.

Since $n < m'$, 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 any vector $v \approx sG$ it is easy to find s, making it possible for the adversary to recover the secret keys of the honest parties. (This is exactly where the "dual" scheme helps: the adversary still sees some "leakage" $v \approx sA_N$, 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 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, $A^T = (A_1^T|\ldots|A_N^T)$. 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 bottom few rows in AR are equal to the top few rows, which lets the adversary distinguish AR from a uniform random U.

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. For example, 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.)