Two Round Multiparty Computation via Multi-Key FHE

Pratyay Mukherjee[*] Daniel Wichs[†]

February 27, 2016

Abstract

We construct a general multiparty computation (MPC) protocol with only two rounds of interaction in the common random string model, which is known to be optimal. In the honest-but-curious setting we only rely on the learning with errors (LWE) assumption, and in the fully malicious setting we additionally assume the existence of non-interactive zero knowledge arguments (NIZKs). Previously, Asharov et al. (EUROCRYPT '12) showed how to achieve three rounds based on LWE and NIZKs, while Garg et al. (TCC '14) showed how to achieve the optimal two rounds based on indistinguishability obfuscation, but it was unknown if two rounds were possible under standard assumptions without obfuscation.

Our approach relies on *multi-key fully homomorphic encryption (MFHE)*, introduced by Lopez-Alt et al. (STOC '12), which enables homomorphic computation over data encrypted under different keys. We present a construction of MFHE based on LWE that significantly simplifies a recent scheme of Clear and McGoldrick (CRYPTO '15). We then extend this construction to allow for a one-round distributed decryption of a multi-key ciphertext. Our entire MPC protocol consists of the following two rounds:

1. Each party individually encrypts its input under its own key and broadcasts the ciphertext. All parties can then homomorphically compute a multi-key encryption of the output.
2. Each party broadcasts a partial decryption of the output using its secret key. The partial decryptions can be combined to recover the output in plaintext.

[*] University of California, Berkeley. E-mail: `pratyay85@gmail.com`. Research supported in part from DARPA Safeware Award W911NF15C0210, AFOSR Award FA9550-15-1-0274 and NSF CRII Award 1464397. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government. Part of the work was done when this author was a PhD Fellow at Aarhus University supported by ERC starting grant 279447, the CTIC and CFEM research center (under the Sino-Danish grant no. 61061130540)

[†] Northeastern University. E-mail: `wichs@ccs.neu.edu`. Research supported by NSF grants CNS-1347350, CNS-1314722, CNS- 1413964.

Contents

1 Introduction

Multiparty Computation. Secure multiparty computation (MPC) allows multiple parties to evaluate an arbitrary function over their inputs *privately*, without revealing anything about their inputs to each other beyond the function's output. This problem was initially studied by Yao [Yao82, Yao86], in the case of *two honest-but-curious parties* (who follow the protocol honestly but hope to learn information from its execution) and later by Goldreich, Micali and Wigderson [GMW87] in the case of an arbitrary number of *fully malicious* parties (who can deviate arbitrarily from the specified protocol execution). By now, MPC is a fundamental part of cryptography and a subject of intense study.

One of the main challenges is to optimize the efficiency of MPC protocols. In this work, our main focus will be on constructing MPC protocols with the optimal round complexity. [1]

Round Complexity of MPC. We refer the reader to [AJLA$^+$12] for a comprehensive overview of prior work on round complexity of MPC. In the honest-but-curious setting, it was known how to achieve a constant number of rounds assuming the existence of oblivious transfer [BMR90, IK00, Lin01, KOS03, AIK05]. However, the concrete constants were not explicitly stated and they seem to require at least 4 rounds. These protocols can also be compiled into secure constructions in the fully malicious setting with only a constant number of additional rounds by using coin-flipping and concurrent zero-knowledge proofs [Lin01, KOS03, Goy11, LP11]. In the plain model and the fully malicious setting, there is a known lower bounds of 5 rounds for two party computation[2], albeit in non-simultaneous message model where no broadcast channel is available [KO04]. A recent work [GMPP15] shows a similar lower bound of 4 rounds assuming broadcast channel. However, in the honest-but-curious setting or even in the fully malicious setting with a common random string (CRS) the above lower bound does not hold and there is only a simple lower bound of 2 rounds [HLP11]. In this work, we will assume the CRS model.

Recently, a result of Asharov et al. [AJLA$^+$12] showed how to achieve a 3 round MPC protocol in the CRS model, by relying on techniques from fully homomorphic encryption (FHE). Their construction achieves semi-honest security under the *learning with errors* (LWE) assumption, and fully malicious security (in the universal composability (UC) framework) by further assuming the existence of non-interactive zero knowledge arguments (NIZKs). The construction also yields a 2 round protocol in the public-key infrastructure (PKI) model, but it was left as an open problem to achieve 2 rounds in the CRS model.

Even more recently, the results of Garg et al. [GGHR14, GP15] achieve a 2 round MPC protocol in the CRS model by relying on *indistinguishability obfuscation* (iO) and statistically sound NIZKs. On a high level, the main idea of that work is to have each party obfuscates its "next-message" function, after an initial round where the parties commit to their input. Making this work under the iO assumption is non-trivial and requires much care. However, this approach appears to crucially rely on obfuscation and does not easily lend itself to instantiations under simpler assumptions.

[1] We assume a broadcast communication channel and in each round of a protocol all parties broadcast a message to all other parties. Each honest party must broadcast their round i message right away prior to receiving the round i messages of other parties. On the other hand, we assume a "rushing" adversary that can wait to collect the round i messages of all honest parties prior to selecting the round i messages of the corrupted parties.

[2] They consider a model where broadcast channel is not present, in that only one party sends a message to another and in a round parties are not allowed to send messages simultaneously.

The main open question left by these works is whether 2 round MPC is achievable under more "standard" cryptographic assumptions, without relying on obfuscation.

Our Result. In this work, we construct a 2 round MPC protocol in the CRS model. We achieve honest-but-curious security under only the LWE assumption, and fully malicious security (in the UC framework) by additionally assuming the existence of NIZKs. As our main technical result, which may be of independent interest, we show how to construct a multi-key fully homomorphic encryption scheme with a one-round threshold decryption protocol.

2 Overview of Our Techniques

We now give an overview of our techniques by first describing how to construct MPC from multi-key FHE with threshold decryption, and then how to construct the latter from LWE.

2.1 MPC via Threshold (Multi-Key) FHE

MPC via Threshold FHE. We begin with the approach of Asharov et al. [AJLA$^+$12] (variants of which were used in many preceding works [FH96, JJ00, CDN01, DN03, BD10, BDOZ11, MSS11]) for constructing MPC based on fully homomorphic encryption (FHE). At a high level, this approach is based on the following simple template:

1. The parties first run a secure distributed protocol for the "threshold key-generation" of an FHE scheme to agree on a common public key *pk* and a secret sharing of the corresponding secret key *sk* so that each party holds one share, and all shares are needed to recover *sk*.

2. Each party i then broadcasts an encryption of its input x_i under the common public key *pk*. Note that no individual party or incomplete set of parties can decrypt this ciphertext and so the privacy of the input is maintained. At the end of this round, each party can homomorphically compute the desired function f on the received ciphertexts and derive a common output ciphertext which encrypts $y = f(x_1, \ldots, x_N)$.

3. The parties run a secure distributed protocol for "threshold decryption" using their shares of the secret key *sk* to decrypt the output ciphertext and recover the output y in plaintext.[3]

Secure protocols for threshold key-generation and decryption can be implemented generically for any FHE scheme by using general MPC techniques, but this would require many rounds. Instead [AJLA$^+$12] show that specific FHE schemes by Brakerski, Gentry and Vaikuntanathan [BV11, BGV12] based on the LWE assumption have a "key homomorphic" property which can be leveraged to get distributed key-generation and decryption protocols consisting of one round each. Therefore, when instantiated with these schemes, the above template results in a 3 round MPC protocol.[4]

[3] Throughout this work, we use the term "threshold" to denote distributed schemes where *all* parties are needed to perform an operation and security is maintained from any incomplete subset of parties.

[4] We note that one the main challenges in the work of [AJLA$^+$12] is to implement the threshold generation of the FHE "evaluation key" which has a complex structure in the FHE schemes of [BV11, BGV12]. This could be vastly simplified using a more recent FHE scheme of Gentry-Sahai-Waters [GSW13] which does not require an evaluation key. However, this would still not improve the final round complexity of the MPC construction below 3 rounds.

MPC via Threshold Multi-Key FHE. One could hope to shave off an additional round from the above template by using *multi-key fully homomorphic encryption* (MFHE), recently introduced by Lopez-Alt, Tromer and Vaikuntanathan [LTV12]. An MFHE schemes allows parties to independently encrypt their data under different individually chosen keys, while still allowing homomorphic computations over such ciphertexts. The output of such homomorphic computation is a "multi-key ciphertext" which cannot be decrypted by any single party individually (as this would violate semantic security of the other parties) but can be decrypted by the parties jointly using the combination of all their secret keys. The work of [LTV12] constructed such an MFHE scheme based on (a variant of) the NTRU assumption.

Using MFHE, we naturally get the following simplified template for MPC:

1. Each party individually chooses its own MFHE key pair (pk_i, sk_i), encrypts its input x_i under pk_i, and broadcasts the resulting ciphertext. At the end of this round, each party can homomorphically compute the desired function f on the received ciphertexts and derive a common multi-key ciphertext which encrypts the output $y = f(x_1, \ldots, x_N)$.

2. The parties run a secure distributed protocol for "threshold decryption" using their secret keys sk_i to decrypt the multi-key ciphertext and recover the output y in plaintext.

As before, a distributed threshold decryption can be implemented generically using general MPC techniques, but this would require many rounds. Unfortunately, the MFHE scheme of [LTV12] does not appear to admit any simpler threshold decryption protocol and therefore it is not known how to use this scheme to get a 2 round MPC.

A recent work of Clear and McGoldrick [CM15] gives an alternate construction of MFHE based on the LWE assumption, by cleverly adapting an FHE scheme of Gentry, Sahai and Waters [GSW13]. We first present a significantly simplified construction of MFHE from LWE and give a stand-alone presentation of this scheme. We then show that this scheme admits a simple 1-round threshold decryption protocol. This threshold decryption protocol only satisfies a weak notion of security which doesn't allow us to directly plug it into the above template for MPC. However, we show that we can make this approach work with only minor additional modifications.

As in [AJLA$^+$12], we show that our basic scheme (based on LWE) achieves security in *semi-malicious* setting, which is a strengthening of the honest-but-curious setting, where parties follow the protocol specification but can choose their random coins adversarially. By using NIZKs, we can then compile such a scheme into one which is secure in the *fully malicious setting* (and even universally composable) without additional rounds.

2.2 Constructing Threshold Multi-Key FHE

We now give a high-level description of the MFHE construction and the threshold decryption protocol. We begin by describing a recent FHE construction by Gentry, Sahai and Waters (GSW) [GSW13] using the notation and exposition of [AP14]. Then describe how to convert it into a MFHE scheme. Finally, we discuss how to perform threshold decryption.

Public Short Preimage Matrix. Before we describe the GSW encryption, we state a useful fact from [MP12] which we heavily rely on in the construction.

Lemma 2.1 ([MP12])**.** *For any $m \geq n\lceil \log q \rceil$ there exists a fixed efficiently computable matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ and an efficiently computable deterministic "short preimage" function $\mathbf{G}^{-1}(\cdot)$ satisfying* *the following. On input a matrix* $\mathbf{M} \in \mathbb{Z}_q^{n \times m'}$ *for any* m'*, the function* $\mathbf{G}^{-1}(\mathbf{M})$ *outputs a bit-matrix* $\mathbf{G}^{-1}(\mathbf{M}) \in \{0,1\}^{m \times m'}$ *such that* $\mathbf{G}\mathbf{G}^{-1}(\mathbf{M}) = \mathbf{M}$.

We can think of $\mathbf{G}$ as a special matrix with a "public trapdoor" that allows us to solve the *short integer solution (SIS)* problem. For those familiar with GSW encryption, multiplication by $\mathbf{G}$ is the $\mathsf{BitDecomp}^{-1}$ operation and the function $\mathbf{G}^{-1}(\cdot)$ is called $\mathsf{BitDecomp}$, but we can ignore the low-level detail of how this is implemented. Note that $\mathbf{G}^{-1}(\cdot)$ is not itself a matrix but rather an efficiently computable function.

Gentry-Sahai-Waters (GSW) FHE. Firstly, choose a random public matrix $\mathbf{B} \in \mathbb{Z}_q^{(n-1) \times m}$ where $m = O(n \log q)$. We can think of this as a common public parameter used by all parties. A public/secret key pair is chosen by selecting a random vector $\mathbf{s} \in \mathbb{Z}_q^{n-1}$ and setting $\mathbf{b} = \mathbf{s}\mathbf{B} + \mathbf{e}$ where $\mathbf{e}$ is some short "error vector". We set the secret key to $\mathbf{t} = (-\mathbf{s}, 1) \in \mathbb{Z}_q^n$ and the public key to the matrix

$$\mathbf{A} := \begin{bmatrix} \mathbf{B} \\ \mathbf{b} \end{bmatrix} \in \mathbb{Z}_q^{n \times m}$$

which ensures that $\mathbf{t}\mathbf{A} = \mathbf{e} \approx \mathbf{0}$ (throughout the introduction, we use $\approx$ to hide "short" values).

A valid GSW ciphertext of a bit $\mu \in \{0,1\}$ with respect to a secret key $\mathbf{t}$ is a matrix $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$ such that $\mathbf{t}\mathbf{C} \approx \mu\mathbf{t}\mathbf{G}$. To encrypt a bit μ using the public key $\mathbf{A}$ we set $\mathbf{C} = \mathbf{A}\mathbf{R} + \mu\mathbf{G}$ where $\mathbf{R} \in \{0,1\}^{m \times m}$ is chosen as a random bit-matrix. This ensures that the result is a valid encryption of μ under the secret key $\mathbf{t}$ since $\mathbf{t}\mathbf{C} = \mathbf{t}\mathbf{A}\mathbf{R} + \mu\mathbf{t}\mathbf{G} \approx \mu\mathbf{t}\mathbf{G}$.

Given two valid GSW ciphertexts $\mathbf{C}_1, \mathbf{C}_2$ encrypting the bits μ_1, μ_2 with respect to a secret key $\mathbf{t}$ we can perform homomorphic addition by setting $\mathbf{C}^+ = \mathbf{C}_1 + \mathbf{C}_2$ and multiplication by setting $\mathbf{C}^\times = \mathbf{C}_1\mathbf{G}^{-1}(\mathbf{C}_2)$. It is a simple exercise to check that $\mathbf{t}\mathbf{C}^+ \approx (\mu_1 + \mu_2)\mathbf{t}\mathbf{G}$ and $\mathbf{t}\mathbf{C}^\times \approx (\mu_1\mu_2)\mathbf{t}\mathbf{G}$. This allows us to homomorphically evaluate any circuit, subject to the error not getting "too large".

Finally, to decrypt a ciphertext $\mathbf{C}$ we set $\mathbf{w} := (0, \ldots, 0, \lceil q/2 \rceil)$ and compute $v = \mathbf{t}\mathbf{C}\mathbf{G}^{-1}(\mathbf{w}^T)$. If $\mathbf{C}$ is a valid encryption of μ under $\mathbf{t}$ then $v \approx \mu\lceil q/2 \rceil$. We recover μ by checking whether v is closer to 0 or to $q/2$.

Multi-Key variant of GSW. We now describe how to convert the above GSW FHE into a multi-key FHE. For simplicity, let's assume that we only have $N = 2$ parties, but everything extends naturally to any polynomial number of parties N. We assume that the matrix $\mathbf{B}$ of the GSW encryption scheme is a common public parameter which is used by all parties.

The two parties choose independent GSW secret keys $\mathbf{t}_1 = (-\mathbf{s}_1, 1), \mathbf{t}_2 = (-\mathbf{s}_2, 1)$ and compute the corresponding public key components $\mathbf{b}_1 = \mathbf{s}_1\mathbf{B} + \mathbf{e}_1$ and $\mathbf{b}_2 = \mathbf{s}_2\mathbf{B} + \mathbf{e}_2$ using the common (and random) $\mathbf{B}$. We let

$$\mathbf{A}_1 := \begin{bmatrix} \mathbf{B} \\ \mathbf{b}_1 \end{bmatrix} \quad , \quad \mathbf{A}_2 := \begin{bmatrix} \mathbf{B} \\ \mathbf{b}_2 \end{bmatrix}$$

be the two GSW public keys for parties 1 and 2 respectively.

Now assume that the two parties independently encrypt some data under their respective keys. Unfortunately, we will not get anything meaningful by naively attempting to perform the GSW homomorphic operations on these ciphertexts under different keys. Instead, our goal will be to first convert both ciphertexts into a "common format" that will allow us to perform homomorphic operations over them.

In particular, we define a "combined secret key" $\widehat{\mathbf{t}} = (\mathbf{t}_1, \mathbf{t}_2) \in \mathbb{Z}_q^{2n}$ as the concatenation of the two individual secret keys. Our goal will be to take a ciphertext $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$ which encrypts a bit μ with respect to the secret key of a single party (along with some helper information specified later) and expand it into multi-key ciphertext $\widehat{\mathbf{C}} \in \mathbb{Z}_q^{2n \times 2m}$ which encrypts μ with respect to the combined secret key $\widehat{\mathbf{t}}$. In particular, a multi-key encryption of a bit μ satisfies $\widehat{\mathbf{t}}\widehat{\mathbf{C}} \approx \mu\widehat{\mathbf{t}}\widehat{\mathbf{G}}$ where $\widehat{\mathbf{G}} = \begin{bmatrix} \mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{G} \end{bmatrix} \in \mathbb{Z}_q^{2n \times 2m}$ is an expanded public matrix with a corresponding short preimage function $\widehat{\mathbf{G}}^{-1}(\cdot)$. Once we do this, we can expand all ciphertexts under individual keys into multi-key ciphertexts under the key $\widehat{\mathbf{t}}$ and then perform homomorphic operations on the multi-key ciphertexts just like in basic GSW scheme (just with larger parameters $n' = 2n, m' = 2m$). Therefore, the only challenge is how to perform the above "ciphertext expansion step".

Ciphertext Expansion. To perform ciphertext expansion, we use a new primitive called "masking scheme" introduced by Clear and McGoldrick in [CM15]. Let $\mathbf{C}$ be a GSW encryption of some bit μ. A masking scheme allows party 1 to create some additional helper information $\mathcal{U}$ about the ciphertext $\mathbf{C}$ at encryption time and release the tuple $(\mathcal{U}, \mathbf{C})$ while keeping the semantic security of the message intact. This information is completely independent of party 2 whose identity is unknown at encryption time. Later, if we are given the public key $\mathbf{A}_2$ for party 2, we can use the information $\mathcal{U}$ to create a matrix $\mathbf{X}$ such that $\mathbf{t}_1\mathbf{X} + \mathbf{t}_2\mathbf{C} \approx \mu\mathbf{t}_2\mathbf{G}$ where $\mathbf{t}_2$ is the secret key of party 2. This allows us to perform ciphertext expansion by creating the expanded ciphertext:

$$\widehat{\mathbf{C}} = \begin{bmatrix} \mathbf{C} & \mathbf{X} \\ \mathbf{0} & \mathbf{C} \end{bmatrix}$$

so that,

$$\widehat{\mathbf{t}}\widehat{\mathbf{C}} \;=\; [\; \mathbf{t}_1\mathbf{C} \;,\; \mathbf{t}_1\mathbf{X} + \mathbf{t}_2\mathbf{C} \;] \quad \approx \quad [\; \mu\mathbf{t}_1\mathbf{G}, \mu\mathbf{t}_2\mathbf{G} \;] = \mu\widehat{\mathbf{t}}\widehat{\mathbf{G}}.$$

We can similarly expand the individually created ciphertexts of party 2 and then perform GSW style homomorphic operations on the expanded ciphertexts.[5] Therefore, the only thing left to do is to construct such a "masking scheme" which we briefly describe below.

A Masking Scheme for GSW. The masking scheme consists of party 1 creating tuple $(\mathcal{U}, \mathbf{C})$ where $\mathbf{C}$ is a GSW encryption of the message μ under its own public key $pk_1 = \mathbf{A}_1$ so that

$$\mathbf{C} := \mathbf{A}_1\mathbf{R} + \mu\mathbf{G} = \begin{bmatrix} \mathbf{B}\mathbf{R} \\ \mathbf{b}_1\mathbf{R} \end{bmatrix} + \mu\mathbf{G}$$

for some random matrix $\mathbf{R} \in \{0,1\}^{m \times m}$. The additional helper information $\mathcal{U}$ consists of m^2 GSW encryptions of each of the scalars $\{\mathbf{R}[a,b]\}_{a \in [m], b \in [m]}$ under the public key pk_1. It is easy to show that the pair $(\mathcal{U}, \mathbf{C})$ computationally hides μ by relying on semantic security of the GSW scheme.

Later, assume we are given the public key $\mathbf{A}_2 := \begin{bmatrix} \mathbf{B} \\ \mathbf{b}_2 \end{bmatrix}$ for party 2, corresponding to a secret key $\mathbf{t}_2 = (-\mathbf{s}_2, 1)$. Then

$$\mathbf{t}_2\mathbf{C} = -\mathbf{s}_2\mathbf{B}\mathbf{R} + \mathbf{b}_1\mathbf{R} + \mu\mathbf{t}_2\mathbf{G} \approx (\mathbf{b}_1 - \mathbf{b}_2)\mathbf{R} + \mu\mathbf{t}_2\mathbf{G}$$

[5] In the actual scheme involving N parties we first expand the single-key ciphertext of each party into a multi-key ciphertext (under the concatenated keys of all the parties) and subsequently perform homomorphic operations on the expanded ciphertexts.

since $\mathbf{b}_2 \approx \mathbf{s}_2\mathbf{B}$. The value $\mathbf{t}_2\mathbf{C}$ corresponds to decrypting the GSW ciphertext $\mathbf{C}$ with the "incorrect" secret key $\mathbf{t}_2$ and it yields the correct value $\mu\mathbf{t}_2\mathbf{G}$ *except* that it is "masked" by the additional term $(\mathbf{b}_1 - \mathbf{b}_2)\mathbf{R}$.

Our goal is to come up with a matrix $\mathbf{X}$ for which $\mathbf{t}_1\mathbf{X} \approx (\mathbf{b}_2 - \mathbf{b}_1)\mathbf{R}$ and therefore adding $\mathbf{t}_1\mathbf{X} + \mathbf{t}_2\mathbf{C} \approx \mu\mathbf{t}_2\mathbf{G}$ as desired. One can do this by homomorphically combining the m^2 ciphertexts contained in $\mathcal{U}$, which encrypt each of the scalars $\mathbf{R}[a, b]$ of the matrix $\mathbf{R}$ under $\mathbf{t}_1$, to get a "pseudo ciphertext" $\mathbf{X}$ which acts like an encryption of the vector $(\mathbf{b}_2 - \mathbf{b}_1)\mathbf{R}$ in the sense that $\mathbf{t}_1\mathbf{X} \approx (\mathbf{b}_2 - \mathbf{b}_1)\mathbf{R}$. This is not a standard homomorphic operation yielding a standard ciphertext – for example, the output is a vector rather than a scalar – but the idea for how to do this is very similar to the way we do standard GSW homomorphic operations. We skip the details of this step in the introduction, and refer the reader to Section 5.1.1 for details.

Threshold Decryption of Multi-Key GSW. A multi-key GSW ciphertext encrypting a bit μ with respect to the expanded secret key $\widehat{\mathbf{t}} = (\mathbf{t}_1, \ldots, \mathbf{t}_N)$ corresponding to N parties, is a matrix $\widehat{\mathbf{C}} \in \mathbb{Z}_q^{nN \times mN}$ such that $\widehat{\mathbf{t}}\widehat{\mathbf{C}} \approx \mu\widehat{\mathbf{t}}\widehat{\mathbf{G}}$.

If we were given all of the secret keys $\widehat{\mathbf{t}} = (\mathbf{t}_1, \ldots, \mathbf{t}_N)$ simultaneously, we could decrypt this ciphertext using the GSW decryption procedure, scaled up to the larger dimension: let $\widehat{\mathbf{w}} = (0, \ldots, 0, \lceil q/2 \rceil) \in \mathbb{Z}_q^{nN}$ and compute $v = \widehat{\mathbf{t}}\widehat{\mathbf{C}}\widehat{\mathbf{G}}^{-1}(\widehat{\mathbf{w}}^T) \approx \mu\lceil q/2 \rceil$.

However, our goal is to design a distributed decryption protocol, where the parties collaboratively decrypt μ without revealing their secret keys to each other. We do this as follows. Let's think of $\widehat{\mathbf{C}}$ as consisting of N matrices $\widehat{\mathbf{C}}^{(i)} \in \mathbb{Z}_q^{n \times mN}$ stacked on top of each other. Then each party i uses its secret key $\mathbf{t}_i$ to output a "partial decryption" $p_i = \mathbf{t}_i\widehat{\mathbf{C}}^{(i)}\widehat{\mathbf{G}}^{-1}(\widehat{\mathbf{w}}^T) + e_i$ where e_i is some "medium-sized smudging error". This error is needed to smudge out any information about the error contained in the ciphertext $\widehat{\mathbf{C}}$, which might contain sensitive information beyond just the plaintext bit. These partial decryptions can be combined to compute $\sum_i p_i \approx v \approx \mu\lceil q/2 \rceil$ and therefore recover the plaintext bit μ.

The above process satisfies the following security notion: given the ciphertext $\widehat{\mathbf{C}}$, the bit μ that it encrypts, and the secret keys $\{\mathbf{t}_i \ : \ i \neq j\}$ of all-but-one of the parties, we can simulate the partial decryption p_j of party j without knowing its secret key $\mathbf{t}_j$. Intuitively, this property says that the partial decryption p_j cannot reveal too much information about $\mathbf{t}_j$.

The above security property of partial decryption is tricky to use since it allows us to simulate the partial decryption of only one party at a time. Nevertheless, we show that this security property of threshold decryption is sufficient in the context of implementing MPC.

2.3 Road-Map Through the Paper

We begin by giving a definition of multi-key FHE (MFHE) first and then MFHE with threshold decryption in Section 4. Then in Section 5 we construct such a scheme from the LWE assumption and in Section 6 we show how to construct MPC from such a scheme. These two sections are independent of each other and can be read in any order. The construction of MFHE with threshold decryption in Section 5 follows in four parts. First, we present the GSW encryption scheme along with a non-standard but useful homomorphic property that it satisfies. Secondly, we define the notion of a masking scheme for GSW and show how to construct it. Thirdly, we use GSW and the masking scheme to construct multi-key FHE. Finally, we show how to perform threshold decryption for such scheme.

3 Preliminaries

Throughout, we let λ denote the *security parameter* and $\mathsf{negl}(\lambda)$ denote a negligible function. We represent elements in $\mathbb{Z}_q$ as integers in the range $(-q/2, q/2]$. Let $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}^n$ be a vector. We use the notation $\mathbf{x}[i]$ to denote the ith component scalar. Similarly for a matrix $\mathbf{M} \in \mathbb{Z}^{n\times m}$ we use $\mathbf{M}[i,j]$ to denote the scalar element located in the i-th row and the j-th column. In general, vectors are represented as single row matrices. The infinity norm (often called simply *norm*) of a vector $\mathbf{x}$ is defined as $\|\mathbf{x}\|_\infty = \max_i(|\mathbf{x}[i]|)$. The norm of matrices is defined similarly. An n-dimensional all-zero vector is usually denoted by $\mathsf{0}^n$ and similarly $\mathsf{0}^{n\times m}$ denotes an all-zero matrix.

For two distributions X, Y, over a finite domain Ω, the *statistical distance* between X and Y is defined by $\Delta(X,Y) \overset{\text{def}}{=} \frac{1}{2}\sum_{\omega\in\Omega}|X(\omega) - Y(\omega)|$. If X, Y are distribution ensembles parameterized by the security parameter, we write $X \overset{\text{stat}}{\approx} Y$ if the quantity $\Delta(X,Y)$ is negligible. Similarly, we write $X \overset{\text{comp}}{\approx} Y$ if they are computationally indistinguishable. We write $\omega \leftarrow X$ to denote that ω is sampled at random according to distribution X. We write $\omega \leftarrow \Omega$ to denote that it is sampled *uniformly at random* from the set Ω. For a distribution ensemble $\chi = \chi(\lambda)$ over the integers, and integers bounds $B = B(\lambda)$, we say that χ is *B-bounded* if $\Pr_{x\leftarrow\chi(\lambda)}[|x| \le B(\lambda)] = 1$.

We rely on the following lemma, which says that adding large noise "smudges out" any small values (see e.g.,[AJW11] for proof).

Lemma 3.1 (Smudging Lemma). *Let $B_1 = B_1(\lambda)$, and $B_2 = B_2(\lambda)$ be positive integers and let $e_1 \in [-B_1, B_1]$ be a fixed integer. Let $e_2 \leftarrow [-B_2, B_2]$ be chosen uniformly at random. Then the distribution of e_2 is* statistically indistinguishable *from that of $e_2 + e_1$ as long as $B_1/B_2 = \mathsf{negl}(\lambda)$.*

Learning With Errors. The *decisional learning with errors (*LWE*)* problem, introduced by Regev [Reg05], is defined as follows.

Definition 3.2 (LWE [Reg05]). *Let λ be the security parameter, $n = n(\lambda), q = q(\lambda)$ be integers and let $\chi = \chi(\lambda)$, be distributions over $\mathbb{Z}$. The $\mathsf{LWE}_{n,q,\chi}$ assumption says that for any polynomial $m = m(\lambda)$ we have*

$$(\mathbf{A}, \mathbf{sA} + \mathbf{e}) \overset{\text{comp}}{\approx} (\mathbf{A}, \mathbf{z})$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$ and $\mathbf{z} \leftarrow \mathbb{Z}_q^m$.

The works of [Reg05, Pei09] show that the LWE problem is as hard as approximating the shortest vector problem in lattices (for appropriate parameters). The version of the LWE assumption that we need here is that for any polynomial $p = p(\lambda)$ there is a polynomial $n = n(\lambda)$, a modulus $q = q(\lambda)$ of singly-exponential size, and a distribution $\chi = \chi(\lambda)$ such that χ is B_χ-bounded and $q \ge 2^p B_\chi$ such that $\mathsf{LWE}_{n,q,\chi}$ holds. This is as hard as approximating the shortest vector with sub-exponential approximation factors.

4 Defining Threshold Multi-Key FHE

4.1 Multi-Key FHE (MFHE)

We start with our definition of (leveled) multi-key FHE which is adapted from the definition given by Lopez-Alt, Tromer and Vaikuntanathan [LTV12] with some minor differences which reflect differences in the properties achieved by the schemes of [LTV12] and [CM15]. On the positive side, in the scheme of [CM15] the number of parties N need not be known ahead of time during key generation or encryption. On the negative side, the scheme of [CM15] requires some common public parameters that are available to the parties during key generation.

Below we call any ciphertext which is associated with multiple keys an "expanded" ciphertext. Also, the ciphertexts that are generated by the encryption procedure (and thus corresponds to a single key) are called "fresh" ciphertexts, and the expanded ciphertexts that are output by the homomorphic evaluations are called "evaluated" ciphertexts.

Definition 4.1. (Multi-key (Leveled) FHE) *A multi-key (leveled) FHE is a tuple of algorithms* $\mathsf{MFHE} = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Encrypt}, \mathsf{Expand}, \mathsf{Eval}, \mathsf{Decrypt})$ *described as follows:*

- $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$*: Setup takes as input the security parameter* λ *and the circuit depth* d *and outputs the system parameters* params*. We assume that all the other algorithms take* params *as an input implicitly.*
- $(sk, pk) \leftarrow \mathsf{Keygen}(\mathsf{params})$*: Output secret key* sk *and public key* pk*.*
- $c \leftarrow \mathsf{Encrypt}(pk, \mu)$*: On input* pk *and some message* μ *output a ciphertext* c*.*
- $\widehat{c} \leftarrow \mathsf{Expand}((pk_1, \ldots, pk_N), i, c)$*: Given a sequence of* N *public-keys and a fresh ciphertext* c *under the* i*-th key* pk_i*, it outputs an "expanded" ciphertext* $\widehat{c}$*.*
- $\widehat{c} := \mathsf{Eval}(\mathsf{params}, \mathcal{C}, (\widehat{c}_1, \ldots, \widehat{c}_\ell))$*: Given a (description of) boolean circuit* $\mathcal{C}$ *of depth* $\leq d$ *along with* ℓ *expanded ciphertexts* $(\widehat{c}_1, \ldots, \widehat{c}_\ell)$*, outputs an evaluated ciphertext* $\widehat{c}$*.*
- $\mu := \mathsf{Decrypt}(\mathsf{params}, (sk_1, \ldots, sk_N), c)$*: On input some ciphertext* $\widehat{c}$ *and a sequence of* N *secret keys output a message* μ*.*

We require the following properties:

Semantic security of encryption: *For any polynomial* $d = d(\lambda)$ *and any two messages* μ_0, μ_1 *the following distributions are computationally indistinguishable:*

$$(\mathsf{params}, pk, \mathsf{Encrypt}(pk, \mu_0)) \overset{\text{comp}}{\approx} (\mathsf{params}, pk, \mathsf{Encrypt}(pk, \mu_1))$$

where $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda, 1^d), (sk, pk) \leftarrow \mathsf{Keygen}(\mathsf{params})$.

Correctness and compactness: *Let* $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$*. Consider any sequences of* N *correctly generated key pairs* $\{(pk_i, sk_i) \leftarrow \mathsf{Keygen}(\mathsf{params})\}_{i\in[N]}$ *and any* ℓ*-tuple of messages* $(\mu_1, \ldots, \mu_\ell)$*. For any sequence of indices* $(I_1, \ldots, I_\ell)$ *where each* $I_i \in [N]$ *let* $\{c_i \leftarrow \mathsf{Encrypt}(pk_{I_i}, \mu_i)\}_{i\in[\ell]}$ *be encryptions of the messages* μ_i *under the* I_i*-th public key and let* $\widehat{c}_i \leftarrow \mathsf{Expand}((pk_1, \ldots, pk_N), I_i, c_i)\}_{i\in[\ell]}$ *be the corresponding expanded ciphertexts. Let* $\mathcal{C}$ *be any (boolean) circuit of depth* $\leq d$ *and let* $\widehat{c} := \mathsf{Eval}(\mathcal{C}, (\widehat{c}_1, \ldots, \widehat{c}_\ell)$ *be the evaluated ciphertext. Then the following holds:*

Correctness of Expansion: $\forall\ i\ \in [\ell]\ ,\ \mathsf{Decrypt}((sk_1, \ldots, sk_N), \widehat{c}_i) = \mu_i$.

Correctness of Evaluation: $\mathsf{Decrypt}((sk_1, \ldots, sk_N), \widehat{c}) = \mathcal{C}(\mu_1, \ldots, \mu_\ell)$.

Compactness: *There exists a polynomial* $p(\cdots)$ *such that* $|\widehat{c}| \leq p(\lambda, d, N)$*. In other words the size of* $\widehat{c}$ *should be independent of* $\mathcal{C}$ *and* ℓ*, but can depend on* λ*,* d *and* N*.*

Public-Coin Parameter Generation. By default, we will consider schemes where the Setup algorithm is "public-coin" meaning that its randomness is included in its output. For such algorithms, we can derive params from a common random string.

4.2 Threshold Decryption for MFHE

We now define a multi-key FHE which supports a one-round threshold distributed decryption protocol. Such a protocol consists of two components: (1) given an expanded ciphertext (possibly evaluated) $\widehat{c}$ each party can compute a partial decryption using its secret key sk_i, (2) there is a way to combine the partial decryptions computed by each party to recover the plaintext.

Definition 4.2. *A* Threshold multi-key FHE scheme (TMFHE) *is a multi-key FHE scheme with two additional algorithms* $\mathsf{MFHE.PartDec}, \mathsf{MFHE.FinDec}$ *described as follows:*

- $p_i \leftarrow \mathsf{MFHE.PartDec}(\widehat{c}, (pk_1, \ldots, pk_N), i, sk_i)$*: On input an expanded ciphertext under a sequence of N keys and the i-th secret key output a partial decryption p_i.*

- $\mu \leftarrow \mathsf{MFHE.FinDec}(p_1, \ldots, p_N)$*: On input N partial decryption output the plaintext μ.*

Along with the properties of multi-key FHE we require the scheme to satisfy the following properties.

Correctness and Simulation: *Let* $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$*. Consider any sequences of N correctly generated key pairs* $\{(pk_i, sk_i) \leftarrow \mathsf{Keygen}(\mathsf{params})\}_{i\in[N]}$ *and any ℓ-tuple of messages $(\mu_1, \ldots, \mu_\ell)$. For any sequence of indices $(I_1, \ldots, I_\ell)$ where each $I_i \in [N]$ let* $\{c_i \leftarrow \mathsf{Encrypt}(pk_{I_i}, \mu_i)\}_{i\in[\ell]}$ *be encryptions of the messages μ_i under the I_i-th public key and let* $\widehat{c}_i \leftarrow \mathsf{Expand}((pk_1, \ldots, pk_N), I_i, c_i)\}_{i\in[\ell]}$ *be the corresponding expanded ciphertexts. Let $\mathcal{C}$ be any (boolean) circuit of depth $\leq d$ and let* $\widehat{c} := \mathsf{Eval}(\mathcal{C}, (\widehat{c}_1, \ldots, \widehat{c}_\ell)$ *be the evaluated ciphertext.*

Correctness of Decryption: *The following holds with probability 1:*

$$\mathsf{MFHE.FinDec}(\widehat{c}, (p_1, \ldots, p_N)) = \mathcal{C}(\mu_1, \ldots, \mu_\ell)$$

where $\{p_i \leftarrow \mathsf{MFHE.PartDec}(\widehat{c}, (pk_1, \ldots, pk_N), i, sk_i)\}_{i\in[N]}$ *are the partial decryptions.*

Simulatability of partial decryption: *There exists a PPT simulator $\mathcal{S}^{thr}$ which, on input and index $i \in [N]$ and all but the i-th keys $\{sk_j\}_{j\in[N]\setminus\{i\}}$ the evaluated ciphertext $\widehat{c}$ and the output message $\mu := \mathcal{C}(\mu_1, \ldots, \mu_\ell)$ produces a simulated partial decryption $p'_i \leftarrow \mathcal{S}^{thr}(\mu, \widehat{c}, i, \{sk_j\}_{j\in[N]\setminus\{i\}})$ such that*

$$p_i \overset{\text{stat}}{\approx} p'_i$$

where $p_i \leftarrow \mathsf{MFHE.PartDec}(\widehat{c}, (pk_1, \ldots, pk_N), i, sk_i)$*. Note that the randomness is only over the random coins of the simulator and the* $\mathsf{MFHE.PartDec}$ *procedure and all other values are assumed to be fixed (and known).*

The simulatability of partial decryptions property says that we can simulate the partial decryption p_i produced by a single party i given the plaintext value μ and the secret keys of all other parties. Ideally, we would have a stronger definition that allows us to simulate the partial decryptions $\{p_i\}_{i\in S}$ of any subset of the parties S given the secret keys of all other parties (rather than just a single values), but unfortunately we do not know how to achieve this type of security. It turns out that, with a little additional work, the given definition suffices in our MPC construction.

5 Constructing Threshold Multi-Key FHE from LWE

We now show how to construct threshold multi-key FHE from LWE. The construction proceeds in four parts. First, we present the GSW encryption scheme along with a non-standard but useful homomorphic property that it satisfies. Secondly, we define the notion of a masking scheme for GSW and show how to construct it. Thirdly, we use GSW and the masking scheme to construct multi-key FHE. Finally, we show to perform threshold decryption for this scheme.

5.1 GSW Fully Homomorphic Encryption

We now describe the GSW fully homomorphic encryption scheme.

- $\mathsf{params} \leftarrow \mathsf{GSW.SetUp}(1^\lambda, 1^d)$: Choose a lattice dimension parameters $n = n(\lambda, d)$ and B_χ-bounded error distribution $\chi = \chi(\lambda, d)$ and a modulus q of size $q = B_\chi 2^{\omega(d\lambda \log \lambda)}$ such that $\mathsf{LWE}_{n-1,q,\chi,B_\chi}$ holds.[6] Choose $m = n\log(q) + \omega(\log \lambda)$. Finally choose a random matrix $\mathbf{B} \in \mathbb{Z}_q^{n-1\times m}$. Output $\mathsf{params} := (q, n, m, \chi, B_\chi, \mathbf{B})$. We stress that all the other algorithms implicitly get params as input even if we usually do not write this explicitly.
- $\mathsf{GSW.Keygen}(\mathsf{params})$: We separately describe two sub-algorithms to generate secret-key and pubic-key respectively:
  - $\mathsf{GSW.SKGen}(\mathsf{params})$**:** Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^{n-1}$. Output $sk = \mathbf{t} = (-\mathbf{s}, 1) \in \mathbb{Z}_q^n$.
  - $\mathsf{GSW.PKGen}(\mathsf{params}, sk)$: Sample $\mathbf{e} \leftarrow \chi^m$. Set $\mathbf{b} := \mathbf{sB} + \mathbf{e} \in \mathbb{Z}_q^m$. Output $pk = \mathbf{A}$ where, $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is defined as $\mathbf{A} := \begin{bmatrix}\mathbf{B}\\ \mathbf{b}\end{bmatrix}$
- $\mathsf{GSW.Encrypt}(pk, \mu)$**:** Choose a short random matrix as the randomness $\mathbf{R} \xleftarrow{\$} \{0,1\}^{m\times m}$. Then output the encryption of message $\mu \in \{0, 1\}$ as $\mathbf{C} \in \mathbb{Z}_q^{n\times m}$ where,
$$\mathbf{C} := \mathbf{AR} + \mu\mathbf{G}$$
- $\mathsf{GSW.Decrypt}(sk, \mathbf{C})$**:** Let $\mathbf{t} := sk$. Define a vector $\mathbf{w} \in \mathbb{Z}_q^n$ as follows:
$$\mathbf{w} = [0, \ldots, 0, \lceil q/2 \rceil]$$
Then compute $v = \mathbf{tCG}^{-1}(\mathbf{w}^T) \in \mathbb{Z}_q^m$. Finally output $\mu' = \left|\left\lfloor \frac{v}{q/2} \right\rceil\right|$ as the decrypted message.
- On input two ciphertexts $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{Z}_q^{n\times m}$ we can define homomorphic addition, multiplication:
  - $\mathsf{GSW.Add}(\mathbf{C}_1, \mathbf{C}_2)$**:** Output $\mathbf{C}_1 + \mathbf{C}_2 \in \mathbb{Z}_q^{n\times m}$.
  - $\mathsf{GSW.Mult}(\mathbf{C}_1, \mathbf{C}_2)$: Output the matrix product $\mathbf{C}_1\mathbf{G}^{-1}(\mathbf{C}_2) \in \mathbb{Z}_q^{n\times m}$.

  This also allows us to compute a homomorphic NAND gate by outputting $\mathbf{G} - \mathbf{C}_1\mathbf{G}^{-1}(\mathbf{C}_2)$.

We sketch the proof of the following theorem for completeness.

Theorem 5.1 ([GSW13])**.** *The scheme described above is a secure FHE under the* $\mathsf{LWE}_{n-1,q,\chi,B_\chi}$ *assumption.*

[6] The size of q here is bigger than needed for GSW encryption alone in order to support our extensions.

Security. The proof of semantic security consists of two steps. First, we can use the LWE assumption to replace the public key $pk = \mathbf{A}$ with a uniformly random matrix in $\mathbb{Z}_q^{n\times m}$. Then we can use the leftover hash lemma to replace the ciphertext $\mathbf{C} := \mathbf{AR} + \mu\mathbf{G}$ with a uniformly random value $\mathbf{C}'$. We refer the reader to [GSW13] for details.

Correctness. To analyze correctness, it is helpful to define the following notion of a "noisy ciphertext".

Definition 5.2. *(β-noisy ciphertext) A β-noisy ciphertext of some message μ under secret-key $sk = \mathbf{t} \in \mathbb{Z}_q^n$ is a matrix $\mathbf{C} \in \mathbb{Z}_q^{n\times m}$ such that: $\mathbf{tC} = \mu\mathbf{tG} + \mathbf{e}$ for some $\mathbf{e}$ with $\|\mathbf{e}\|_\infty \leq \beta$.*

Encryption: Consider a fresh ciphertext $\mathbf{C} = \mathbf{AR} + \mu\mathbf{G}$ which is generated by encrypting some message μ with some public key $\mathbf{A}$ with corresponding secret key $\mathbf{t}$. First recall that $\mathbf{tA} = \mathbf{e}$ such that $\|\mathbf{e}\|_\infty \leq B_\chi$. Therefore $\mathbf{tC} = \mathbf{e}' + \mu\mathbf{tG}$ where $\mathbf{e}' = \mathbf{eR}$ which implies $\|\mathbf{e}'\|_\infty \leq mB_\chi$. Hence $\mathbf{C}$ is mB_χ-noisy encryption of μ under $\mathbf{t}$. Let us call this value initial noise or $\beta_{init} = mB_\chi$.

Evaluation: Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be two ciphertexts which are β_1 and β_2 noisy encryption of $\mu_1, \mu_2 \in \{0,1\}$ under the key $\mathbf{t}$ respectively, so that: $\mathbf{tC}_1 = \mathbf{e}_1 + \mu_1\mathbf{tG}$ and $\mathbf{tC}_2 = \mathbf{e}_2 + \mu_2\mathbf{tG}$ with $\|\mathbf{e}_1\|_\infty \leq \beta_1, \|\mathbf{e}_2\|_\infty \leq \beta_2$.

- *Addition:* Then their addition will result in a ciphertext $\mathbf{C}^{(+)} = \mathbf{C}_1 + \mathbf{C}_2$ such that, $\mathbf{tC}^{(+)} = \mathbf{e}' + (\mu_1 + \mu_2)\mathbf{tG}$ where $\mathbf{e}' = \mathbf{e}_1 + \mathbf{e}_2$. Clearly this is $\beta_1 + \beta_2$-noisy.
- *Multiplication:* On the other hand the multiplication would produce a ciphertext $\mathbf{C}^{(\times)} = \mathbf{C}_1\mathbf{G}^{-1}(\mathbf{C}_2)$ such that $\mathbf{tC}^{(\times)} = \mathbf{e}'' + \mu_1\mu_2\mathbf{G}$ where $\mathbf{e}'' = \mathbf{eG}^{-1}(\mathbf{C}_2) + \mu_1\mathbf{e}_2$. Clearly $\|\mathbf{e}''\|_\infty \leq (m\beta_1 + \beta_2)$ and the ciphertext $\mathbf{C}^{(\times)}$ is $(m\beta_1 + \beta_2)$-noisy. The same calculation holds for NAND gates.

Decryption: Let $\mathbf{C}$ be a β-noisy encryption of μ so that: $\mathbf{tC} = \mathbf{e} + \mu\mathbf{tG}$ where $\|e\|_\infty = \beta$. Then $v = \mathbf{tCG}^{-1}(\mathbf{w}^T) = e' + \mu(q/2)$ such that $e' = \langle \mathbf{e}\ ,\ \mathbf{G}^{-1}(\mathbf{w}^T)\rangle$. Clearly, $\|e'\|_\infty \leq m\beta$. Now one can observe that decryption works correctly as long as $\|e'\|_\infty < q/4$. Therefore correctness holds as long as $\beta < q/(4m)$. We call this value $\beta_{max} := q/(4m)$.

Consider evaluating a (boolean) circuit of depth d consisting of NAND gates. It takes input fresh ciphertexts (β_{init}-noisy) and each level multiplies the noise by a factor of at most $(m+1)$. Therefore, the final output is β_{final}-noisy ciphertexts where $\beta_{final} = (m+1)^d\beta_{init}$. To ensure correctness of decryption we need $\beta_{final} \leq \beta_{max}$ meaning $B_\chi 4m^2(m+1)^d < q$ which is satisfied by our choice of parameters. This concludes the proof.

5.1.1 Homomorphic Linear Combinations and Pseudo Encryption

We now define an additional homomorphic operation. This operation takes as input GSW ciphertexts $\mathbf{C}_{i,j}$ encrypting the individual entries $\mathbf{M}[i,j]$ of some matrix $\mathbf{M} \in \mathbb{Z}_q^{m\times m}$ under a secret key $\mathbf{t}$. It also takes a plaintext vector $\mathbf{v} \in \mathbb{Z}_q^m$ which specifies the homomorphic function to be computed. The operations outputs a "pseudo ciphertext" $\mathbf{C}_{lc}$ which we can think of as a pseudo encryption of the vector $\mathbf{vM}$, meaning that $\mathbf{tC}_{lc} \approx \mathbf{vM}$. Note that the "pseudo ciphertext" $\mathbf{C}_{lc}$ cannot be correctly decrypted (we can only recover something close to $\mathbf{vM}$ but not the exact value) nor can we further perform any of the standard GSW homomorphic operations on it.

Property 5.3. (Linear combination) *Let* $\mathbf{M} \in \{0,1\}^{m\times m}$ *be a matrix and for* $i \in [m], j \in [m]$ *let* $\mathbf{C}_{i,j} \in \mathbb{Z}_q^{n\times m}$ *be a* β*-noisy GSW encryption of* $\mathbf{M}[i,j]$ *under a secret key* $\mathbf{t} \in \mathbb{Z}_q^n$*. Let* $\mathbf{v} \in \mathbb{Z}_q^m$ *be some vector (not necessarily short). Then there is a polynomial-time deterministic algorithm*

$$\mathbf{C}_{lc} = \mathsf{GSW.LComb}((\mathbf{C}_{1,1},\ldots,\mathbf{C}_{m,m}),\mathbf{v})$$

which outputs $\mathbf{C}_{lc} \in \mathbb{Z}_q^{n\times m}$ *such that* $\mathbf{t}\mathbf{C}_{lc} = \mathbf{v}\mathbf{M} + \mathbf{e}$ *where* $\|\mathbf{e}\|_\infty \le m^3\beta$*.*

Implementation. The algorithm $\mathsf{GSW.LComb}((\mathbf{C}_{1,1},\ldots,\mathbf{C}_{m,m}),\mathbf{v})$ is implemented as follows:

1. For each $i \in [m], j \in [m]$ define a matrix $\mathbf{Z}_{i,j} \in \mathbb{Z}_q^{n\times m}$ as follows:

$$\mathbf{Z}_{i,j}[a,b] := \begin{cases} \mathbf{v}[i] & \text{when } a = n \text{ and } b = j \\ 0 & \text{otherwise} \end{cases}$$

   In other words $\mathbf{Z}_{i,j}$ will have 0 everywhere except the n-th (final) row and j-th column where it has the value $\mathbf{v}[i]$.

2. Now output $\mathbf{C}_{lc} \in \mathbb{Z}_q^{n\times m}$ where: $\mathbf{C}_{lc} = \displaystyle\sum_{i=1,j=1}^{m,m} \mathbf{C}_{i,j}\mathbf{G}^{-1}(\mathbf{Z}_{i,j})$

Correctness. Correctness follows because,

$$\begin{aligned}
\mathbf{t}\mathbf{C}_{lc} &= \mathbf{t}\sum_{i,j}\mathbf{C}_{i,j}\mathbf{G}^{-1}(\mathbf{Z}_{i,j}) \\
&= \sum_{i,j}(\mathbf{M}[i,j]\mathbf{t}\mathbf{G} + \mathbf{e}_{i,j})\mathbf{G}^{-1}(\mathbf{Z}_{i,j}) \\
&= \sum_{i,j}(\mathbf{M}[i,j]\mathbf{t}\mathbf{Z}_{i,j} + \mathbf{e}'_{i,j}) \\
&= \mathbf{t}\sum_{i,j}\mathbf{M}[i,j]\mathbf{Z}_{i,j} + \sum_{i,j}\mathbf{e}'_{i,j} \\
&= (-\mathbf{s},1)\begin{bmatrix} 0^{n-1} \\ \mathbf{v}\mathbf{M} \end{bmatrix} + \mathbf{e}'' = \mathbf{v}\mathbf{M} + \mathbf{e}''
\end{aligned}$$

where $\mathbf{e}_{i,j}$ is the noise contained in $\mathbf{C}_{i,j}$ which is of magnitude $\|\mathbf{e}_{i,j}\|_\infty \le \beta$, $\mathbf{e}'_{i,j} = \mathbf{e}_{i,j}\mathbf{G}^{-1}(\mathbf{Z}_{i,j})$ has magnitude $\|\mathbf{e}_{i,j}\|_\infty \le m\beta$, and finally $\mathbf{e}'' = \displaystyle\sum_{i,j}\mathbf{e}'_{i,j}$ has magnitude $\|\mathbf{e}''\|_\infty \le m^3\beta$.

5.2 A Masking Scheme for GSW

We now define and show how to construct a "masking scheme" for GSW, which serves as the main component of the multi-key FHE scheme. Intuitively, a masking scheme allows us to take a GSW public key $pk = \mathbf{A}$ (having a corresponding secret key $\mathbf{t}$) and a bit μ and output a pair of values $(\mathcal{U},\mathbf{C})$ such that $\mathbf{C}$ is a GSW encryption of μ with pk and $\mathcal{U}$ is an auxiliary value such that (1) the pair $(\mathcal{U},\mathbf{C})$ computationally hide μ (just like $\mathbf{C}$ alone) and (2) later, given another GSW public key $pk = \mathbf{A}'$ (having a corresponding secret key $\mathbf{t}'$) we can compute a matrix $\mathbf{X}$ such that $\mathbf{t}\mathbf{X} + \mathbf{t}'\mathbf{C} = \mu\mathbf{t}'\mathbf{G}$.

Property 5.4. (GSW Masking Scheme) *There exists a pair of algorithms* $(\mathsf{UniEnc}, \mathsf{Extend})$*:*

- $\mathsf{UniEnc}(\mu, pk)$*: On input a message* $\mu \in \{0,1\}$ *and a GSW public key* pk *it generates a pair* $(\mathcal{U}, \mathbf{C})$ *where* $\mathbf{C} \in \mathbb{Z}_q^{n\times m}$ *and* $\mathcal{U} \in \{0,1\}^*$*.*
- $\mathsf{Extend}(\mathcal{U}, pk, pk')$*: On input* $\mathcal{U}$ *and GSW public keys* pk, pk' *it outputs* $\mathbf{X} \in \mathbb{Z}_q^{n\times m}$*.*

for which the following properties holds:

Semantic Security: *For any polynomial* $d = d(\lambda)$ *security of GSW encryption implies that:*

$$(\mathsf{params}, pk, \mathsf{UniEnc}(0, pk)) \overset{\text{comp}}{\approx} (\mathsf{params}, pk, \mathsf{UniEnc}(1, pk))$$

where $\mathsf{params} \leftarrow \mathsf{GSW.SetUp}(1^\lambda, 1^d)$*,* $(sk, pk) \leftarrow \mathsf{GSW.Keygen}(\mathsf{params})$*.*

Correctness: *Let* $\mathsf{params} \leftarrow \mathsf{GSW.SetUp}(1^\lambda, 1^d)$ *and let* $(sk = \mathbf{t}, pk), (sk' = \mathbf{t}', pk')$ *be two independent key pairs generated with* $\mathsf{GSW.Keygen}(\mathsf{params})$*. For any* $\mu \in \{0,1\}$ *let* $(\mathcal{U}, \mathbf{C}) \leftarrow \mathsf{UniEnc}(\mu, pk)$ *and* $\mathbf{X} \leftarrow \mathsf{Extend}(\mathcal{U}, pk, pk')$*. Then*

$$\mu := \mathsf{GSW.Decrypt}(sk, \mathbf{C}) \quad \textit{and} \quad \mathbf{tX} + \mathbf{t}'\mathbf{C} = \mu\mathbf{t}'\mathbf{G} + \mathbf{e}$$

where $\|\mathbf{e}\|_\infty \leq \beta_{mask}$ *for* $\beta_{mask} := (m^4 + m)B_\chi$*.*

Instantiation. We now show how to implement such masking scheme.

- $\mathsf{UniEnc}(pk, \mu)$: On input a message μ and a public key pk the algorithm outputs $\mathcal{U}$, which is a m^2-tuple of matrices in $\mathbb{Z}_q^{n\times m}$, and $\mathbf{C} \in \mathbb{Z}_q^{n\times m}$ as follows.
  1. Let $\mathbf{A} = pk$. Set $\mathbf{C} \leftarrow \mathsf{GSW.Encrypt}(pk, \mu) \in \mathbb{Z}_q^{n\times m}$ so that $\mathbf{C} = \mathbf{AR} + \mu\mathbf{G}$ where $\mathbf{R} \in \{0,1\}^{m\times m}$ is the encryption randomness.
  2. Encrypt each element of the random matrix $\mathbf{R}$ (chosen in Step 1) to get m^2 ciphertexts: $\mathbf{V}^{(a,b)} \leftarrow \mathsf{GSW.Encrypt}(pk, \mathbf{R}[a,b])$. Set $\mathcal{U} := \big(\mathbf{V}^{(1,1)}, \ldots, \mathbf{V}^{(m,m)}\big) \in \big(\mathbb{Z}_q^{n\times m}\big)^{(m^2)}$.
- $\mathsf{Extend}(\mathcal{U}, pk, pk')$**:** On input a $\mathcal{U} \in \big(\mathbb{Z}_q^{n\times m}\big)^{(m^2)}$ and public keys pk, pk' the algorithm computes $\mathbf{X} \in \mathbb{Z}_q^{n\times m}$ as follows:
  1. Parse $pk = \mathbf{A} = \begin{bmatrix}\mathbf{B}\\ \mathbf{b}\end{bmatrix}$, $pk' = \mathbf{A}' = \begin{bmatrix}\mathbf{B}\\ \mathbf{b}'\end{bmatrix}$ and, $\mathcal{U} = \big(\{\mathbf{V}^{(a,b)}\}_{a,b\in[m]}\big)$.
  2. Set $\mathbf{X} = \mathsf{GSW.LComb}\big((\mathbf{V}^{(1,1)}, \ldots, \mathbf{V}^{(m,m)}), \mathbf{b}' - \mathbf{b}\big)$.

Semantic Security. The view of the attacker is the following distribution:

$$\Big(\mathsf{params}, \mathbf{A}, \mathbf{C}, \mathcal{U} = \big(\ \mathbf{V}^{(11)}, \ldots, \mathbf{V}^{(m,m)}\big)\Big)$$

generated via $\mathsf{params} \leftarrow \mathsf{GSW.SetUp}(1^\lambda, 1^d)$, $(sk, pk = \mathbf{A}) \leftarrow \mathsf{GSW.Keygen}(\mathsf{params})$ and $(\mathbf{C}, \mathcal{U}) \leftarrow \mathsf{UniEnc}(pk, \mu)$, where either $\mu = 0$ or $\mu = 1$. We prove semantic security of the masking scheme by relying on the semantic security of the underlying GSW scheme. The proof consists of the following hybrids:

- Firstly, we modify each of the ciphertexts $\mathbf{V}^{(a,b)}$ so that instead of being GSW encryptions of $\mathbf{R}[a,b]$, we just choose them as GSW encryptions of 0. This just relies on semantic security of GSW encryption.

- Secondly, we also choose $\mathbf{C}$ as a GSW encryption of 0. This also just follows from the semantic security of GSW encryption, since after the first step no information about the randomness $\mathbf{R}$ is given out.

Finally, this distribution is completely independent of the bit μ which concludes the proof of semantic security.

Correctness. Let $((sk = \mathbf{t}, pk = \mathbf{A}), (sk' = \mathbf{t}', pk' = \mathbf{A}'))$ be two correctly generated GSW key-pairs. Now recall that, $sk = \mathbf{t} = (-\mathbf{s}, 1) \in \mathbb{Z}_q^n$, and $sk' = \mathbf{t}' = (-\mathbf{s}', 1) \in \mathbb{Z}_q^n$; $pk = \mathbf{A} = \begin{bmatrix} \mathbf{B} \\ \mathbf{b} \end{bmatrix} \in \mathbb{Z}_q^{n \times m}$, $pk' = \mathbf{A}' = \begin{bmatrix} \mathbf{B} \\ \mathbf{b}' \end{bmatrix} \in \mathbb{Z}_q^{n \times m}$ where $\mathbf{b} = \mathbf{s}\mathbf{B} + \mathbf{e}$, $\mathbf{b}' = \mathbf{s}'\mathbf{B} + \mathbf{e}'$ with $\|\mathbf{e}\|_\infty, \|\mathbf{e}'\|_\infty \leq \beta_\chi$.

Furthermore, for any message μ let $(\mathcal{U}, \mathbf{C}) \leftarrow \mathsf{UniEnc}(pk, \mu)$ and $\mathbf{X} \leftarrow \mathsf{Extend}(\mathcal{U}, pk, pk')$ where $\mathcal{U} = \left(\mathbf{V}^{(1,1)}, \ldots, \mathbf{V}^{(nm)}\right)$. Then it is easy to see that $\mu := \mathsf{GSW.Decrypt}(sk, \mathbf{C})$ which implies that $\mathbf{C} = \mathbf{A}\mathbf{R} + \mu\mathbf{G} = \begin{bmatrix} \mathbf{B} \\ \mathbf{b} \end{bmatrix} \mathbf{R} + \mu\mathbf{G}$ for some $\mathbf{R} \in \{0,1\}^{m \times m}$ and hence

$$\begin{aligned} \mathbf{t}'\mathbf{C} &= (-\mathbf{s}', 1) \begin{bmatrix} \mathbf{B} \\ \mathbf{b} \end{bmatrix} \mathbf{R} + \mu\mathbf{t}'\mathbf{G} \\ &= -\mathbf{s}'\mathbf{B}\mathbf{R} + \mathbf{b}\mathbf{R} + \mu\mathbf{t}'\mathbf{G} \\ &= -(\mathbf{b}' - \mathbf{e}')\mathbf{R} + \mathbf{b}\mathbf{R} + \mu\mathbf{t}'\mathbf{G} \\ &= (\mathbf{b} - \mathbf{b}')\mathbf{R} + \mu\mathbf{t}'\mathbf{G} + \mathbf{e}_C \end{aligned}$$

where $\mathbf{e}_C = \mathbf{e}'\mathbf{R}$ has norm $\|\mathbf{e}_C\|_\infty = mB_\chi$.

On the other hand, by the correctness of linear combinations, we have:

$$\mathbf{t}\mathbf{X} = (\mathbf{b}' - \mathbf{b})\mathbf{R} + \mathbf{e}_X$$

where $\|\mathbf{e}_X\|_\infty = m^4 B_\chi$.

Combining these equations, we get $\mathbf{t}\mathbf{X} + \mathbf{t}'\mathbf{C} = \mu\mathbf{t}'\mathbf{G} + \mathbf{e}^*$ where $\|\mathbf{e}^*\|_\infty \leq (m^4 + m)B_\chi$ as claimed.

5.3 Construction of Multi-Key FHE

First recall the fixed matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ that played an important role for the earlier construction and analysis. In this section we define an "expanded matrix" $\widehat{\mathbf{G}}_N \in \mathbb{Z}_q^{nN \times mN}$ as:

$$\widehat{\mathbf{G}}_N = \begin{bmatrix} \mathbf{G} & \cdots & \cdots & 0 \\ 0 & \mathbf{G} & \cdots & \vdots \\ \vdots & \cdots & \mathbf{G} & 0 \\ 0 & \cdots & \cdots & \mathbf{G} \end{bmatrix}$$

We note that there exists a corresponding efficiently computable function $\widehat{\mathbf{G}}_N^{-1}(\cdot)$ such that for any $m' \in \mathbb{N}$ any matrix $\mathbf{M} \in \mathbb{Z}_q^{nN \times m'}$, $\widehat{\mathbf{G}}_N^{-1}(\mathbf{M}') \in \{0,1\}^{mN \times mN}$ is "short " and $\widehat{\mathbf{G}}_N \widehat{\mathbf{G}}_N^{-1}(\mathbf{M}) = \mathbf{M}$. Such $\widehat{\mathbf{G}}_N^{-1}(\cdot)$ can be computed using $\mathbf{G}^{-1}(\cdot)$ in the natural way.

Construction. Now we describe our multi-key FHE construction.

- $\mathsf{MFHE.SetUp}(1^\lambda, 1^d)$**:** Run the set-up algorithm of GSW to generate the parameters:

$$\mathsf{params} := (q, n, m, \chi, B_\chi, \mathbf{B}) \leftarrow \mathsf{GSW.SetUp}(1^\lambda, 1^d).$$

- $\mathsf{MFHE.Keygen}(\mathsf{params})$**:** Run the key-generation algorithm of GSW to generate:

$$sk := \mathbf{t} \leftarrow \mathsf{GSW.SKGen}(\mathsf{params}) \qquad pk := \mathbf{A} \leftarrow \mathsf{GSW.PKGen}(\mathsf{params}, sk)$$

- $\mathsf{MFHE.Encrypt}(pk, \mu)$**:** Execute the following steps:
  - Just use the masking scheme: $(\mathcal{U}, \mathbf{C}) \leftarrow \mathsf{UniEnc}(\mu, pk)$.
  - Output the pair $c := (\mathcal{U}, \mathbf{C})$ as the ciphertext for μ.

- $\mathsf{MFHE.Expand}((pk_1, \ldots, pk_N), i, c)$: On receiving a sequence of public-keys $(pk_1, \ldots, pk_N)$ and a fresh ciphertext $c = (\mathcal{U}, \mathbf{C})$ under the public key pk_i run the Extend algorithm for all pk_j where $i \neq j$.
  - For $j \in \{pk_1, \ldots, pk_N\} \setminus \{i\}$, compute $\mathbf{X}_j \leftarrow \mathsf{Extend}(\mathcal{U}, pk_i, pk_j)$.
  - Then define a matrix $\widehat{\mathbf{C}} \in \mathbb{Z}_q^{nN \times mN}$ as a concatenation of N^2 sub-matrices where each sub-matrix $\mathbf{C}_{a,b} \in \mathbb{Z}_q^{n \times m}$ for $a, b \in [N]$ is defined as:

$$\mathbf{C}_{a,b} := \begin{cases} \mathbf{C} & \text{when } a = b \\ \mathbf{X}_j & \text{when } a = i \neq j \text{ and } b = j \\ \mathbf{0}^{n \times m} & \text{otherwise} \end{cases}$$

  For reader's convenience we provide a pictorial representation of $\widehat{\mathbf{C}}$ in Fig. 1:
  Finally output $\widehat{c} := \widehat{\mathbf{C}}$ as the expanded ciphertext.

- $\mathsf{MFHE.Eval}(\mathsf{params}, \mathcal{C}, (\widehat{c}_1, \ldots, \widehat{c}_\ell))$ On input ℓ expanded ciphertexts simply use the GSW homomorphic evaluation algorithms namely $\mathsf{GSW.Add}$ and $\mathsf{GSW.Mult}$, albeit with expanded dimensions $n' = nN$ and $m' = mN$ and the expanded $\widehat{\mathbf{G}}_N, \widehat{\mathbf{G}}_N^{-1}$ (in place of n, m and $\mathbf{G}, \mathbf{G}^{-1}$).

- $\mathsf{MFHE.Decrypt}(\mathsf{params}, (sk_1, \ldots, sk_N), c)$: On input a ciphertext $c = \widehat{\mathbf{C}}$ and the sequence of secret keys $(sk_1, \ldots, sk_N)$ parse each $\mathbf{t}_i := sk_i$ and then construct the joint secret key by horizontally appending all the secret-keys in sequence $\widehat{\mathbf{t}} = \begin{bmatrix} \widehat{\mathbf{t}}_1 & \widehat{\mathbf{t}}_2 & \cdots & \widehat{\mathbf{t}}_N \end{bmatrix} \in \mathbb{Z}_q^{nN}$. Then run the GSW decryption algorithm albeit with expanded dimensions $n' = nN$ and $m' = mN$ and the expanded $\widehat{\mathbf{G}}_N, \widehat{\mathbf{G}}_N^{-1}$ (in place of n, m and $\mathbf{G}, \mathbf{G}^{-1}$).

$$\begin{array}{r}
\\ \\ \\ \text{Row } i \longrightarrow \\ \\ \\ \\ \\
\end{array}
\begin{array}{c}
\begin{bmatrix}
\mathbf{C} & 0 & \cdots & 0 & 0 \\
0 & \ddots & \cdots & \cdots & \vdots \\
\vdots & 0 & 0 & \cdots & 0 \\
\mathbf{X}_1 & \cdots & \mathbf{C} & \cdots & \mathbf{X}_N \\
0 & \cdots & 0 & 0 & \vdots \\
\vdots & \cdots & \cdots & \ddots & 0 \\
0 & 0 & \cdots & 0 & \mathbf{C}
\end{bmatrix} \\
\begin{array}{ccccc} \phantom{\mathbf{C}} & & \uparrow & & \phantom{\mathbf{C}} \\ & & \text{Column } i & & \end{array}
\end{array}$$

Figure 1: Structure of the expanded ciphertext $\widehat{\mathbf{C}}$

5.3.1 Correctness and Security of MFHE Construction

Theorem 5.5. *The scheme described above is a secure MFHE under the* $\mathsf{LWE}_{n-1,q,\chi,B_\chi}$ *assumption (with the same parameters as we defined for GSW encryption).*

Semantic security. The semantic security of the above multi-key FHE follows directly from that of the GSW masking scheme.

Correctness of Expansion. Consider a sequences of N correctly generated key pairs $((sk_1 = \mathbf{t}_1, pk_1), \ldots, (sk_N = \mathbf{t}_N, pk_N))$ such that $\{(pk_i, sk_i) \leftarrow \mathsf{MFHE.Keygen}(\mathsf{params})\}_{i\in[N]}$. Now suppose for any message μ and any $i \in [N]$ we have a ciphertext $c \leftarrow \mathsf{MFHE.Encrypt}(pk_i, \mu)$ under the i-th key and the corresponding expanded ciphertext $\widehat{\mathbf{C}} \leftarrow \mathsf{MFHE.Expand}((pk_1, \ldots, pk_N), i, c)$ as shown in Fig. 1. Let $\widehat{\mathbf{t}} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]$. Then

$$\begin{aligned}
\widehat{\mathbf{t}}\widehat{\mathbf{C}} &= [\mathbf{t}_i\mathbf{X}_1 + \mathbf{t}_1\mathbf{C}, \ldots, \mathbf{t}_i\mathbf{C}, \ldots, \mathbf{t}_i\mathbf{X}_N + \mathbf{t}_N\mathbf{C}] \\
&= [\mu\mathbf{t}_1\mathbf{G} + \mathbf{e}_1, \ldots, \mu\mathbf{t}_i\mathbf{G} + \mathbf{e}_i, \ldots, \mu\mathbf{t}_N\mathbf{G} + \mathbf{e}_N] \\
&= \mu\widehat{\mathbf{t}}\widehat{\mathbf{G}} + [\mathbf{e}_1, \ldots, \mathbf{e}_N]
\end{aligned}$$

where $\|\mathbf{e}_i\|_\infty \leq mB_\chi$ by the correctness of GSW encryption and for $j \neq i$, $\|\mathbf{e}_j\|_\infty \leq (m^4+m)B_\chi$ by the correctness of the GSW masking scheme. Therefore, $\widehat{\mathbf{t}}\widehat{\mathbf{C}} = \mu\widehat{\mathbf{t}}\widehat{\mathbf{G}} + \mathbf{e}$ where $\|\mathbf{e}\|_\infty \leq (m^4+m)B_\chi$. Let's call this value $\beta'_{init} = (m^4 + m)B_\chi = 2^{O(\log\lambda)}B_\chi$. The correctness of GSW encryption is guaranteed as long as $\beta'_{init} \leq q/(4m')$ which holds with the choice of q we defined.

Correctness of Evaluation. Let $\widehat{\mathbf{C}}_1, \ldots, \widehat{\mathbf{C}}_\ell$ be expanded ciphertexts corresponding to bit $\mu_1, \ldots, \mu_\ell$ so that, by the above correctness property, $\widehat{\mathbf{t}}\widehat{\mathbf{C}}_i = \mu_i\widehat{\mathbf{t}}\widehat{\mathbf{G}} + \mathbf{e}_i$ where $\|\mathbf{e}_i\|_\infty \leq \beta'_{init}$. If $\widehat{\mathbf{C}}$ is the output of a homomorphic evaluation of a circuit $\mathcal{C}$ of depth d over the above ciphertexts such that $\mu = \mathcal{C}(\mu_1, \ldots, \mu_\ell)$ then by the correctness of GSW homomorphic evaluation with scaled up parameters $n' = nN, m' = mN$ we have $\widehat{\mathbf{t}}\widehat{\mathbf{C}} = \mu\widehat{\mathbf{t}}\widehat{\mathbf{G}} + \mathbf{e}$ where $\|\mathbf{e}\|_\infty \leq \beta'_{init}(m'+1)^d = (m^4+m)B_\chi(mN+1)^d$. Let's call this value $\beta'_{final} = B_\chi(m^4 + m)(mN + 1)^d = 2^{O(d\log\lambda)}B_\chi$. The correctness of GSW encryption is guaranteed as long as $\beta'_{final} \leq q/(4m')$ which holds with the choice of q we defined.

5.4 Threshold Decryption for Multi-key FHE

We now show how to implement threshold decryption for the MFHE construction outlined in the previous section.

MFHE.PartDec($\widehat{c}, (pk_1, \ldots, pk_N), i, sk_i$): On input an expanded ciphertext $\widehat{c} = \widehat{\mathbf{C}} \in \mathbb{Z}_q^{nN \times mN}$ under a sequence of keys $(pk_1, \ldots, pk_N)$ and the i-th secret key $sk_i = \mathbf{t}_i \in \mathbb{Z}_q^n$ do the following:

- Parse $\widehat{\mathbf{C}}$ as consisting of N sub-matrices $\widehat{\mathbf{C}}^{(i)} \in \mathbb{Z}_q^{n \times mN}$ such that
$$\widehat{\mathbf{C}} = \begin{bmatrix} \widehat{\mathbf{C}}^{(1)} \\ \vdots \\ \widehat{\mathbf{C}}^{(N)} \end{bmatrix}.$$
- Define $\widehat{\mathbf{w}} \in \mathbb{Z}_q^{nN}$ as $\widehat{\mathbf{w}} = [0, \ldots, 0, \lceil q/2 \rceil]$.
- Then compute $\gamma_i = \mathbf{t}_i \widehat{\mathbf{C}}^{(i)} \widehat{\mathbf{G}}^{-1}(\widehat{\mathbf{w}}^T) \in \mathbb{Z}_q$ and output $p_i = \gamma_i + e_i^{sm} \in \mathbb{Z}_q$ where $e_i^{sm} \xleftarrow{\$} [-B_{smdg}^{dec}, -B_{smdg}^{dec}]$ is some random "smudging noise" where $B_{smdg}^{dec} = 2^{d\lambda \log \lambda} B_\chi$.

MFHE.FinDec($p_1, \ldots, p_N$): Given $p_1, \ldots, p_N$, compute the sum $p := \sum_{i=1}^N p_i$. Output $\mu := \left| \left\lceil \frac{p}{q/2} \right\rfloor \right|$.

5.4.1 Correctness and Simulation Security

Theorem 5.6. *The above threshold decryption procedures for* MFHE *satisfy correctness and (statistical) simulation security.*

Correctness. Here the entire scheme is same as MFHE except the decryption. So if $\widehat{\mathbf{C}}$ is an evaluated ciphertext encrypting a bit μ and the secret keys are $\widehat{\mathbf{t}} = [\widehat{\mathbf{t}}_1, \ldots, \widehat{\mathbf{t}}_N]$ then, by the analysis used for non-threshold correctness, we have

$$\widehat{\mathbf{t}}\widehat{\mathbf{C}} = \sum_{i \in [N]} \mathbf{t}_i \widehat{\mathbf{C}}^{(i)} = \mu \widehat{\mathbf{t}} \widehat{\mathbf{G}} + \mathbf{e}$$

where $\|\mathbf{e}\|_\infty \le \beta'_{final} = (m^4 + m) B_\chi (mN+1)^d$. Therefore if the partial decryptions p_i are computed as specified we have:

$$\begin{aligned} \sum_{i \in [N]} p_i &= \sum_{i \in [N]} \gamma_i + \sum_{i \in [N]} e_i^{sm} = \sum_{i \in [N]} \mathbf{t}_i \widehat{\mathbf{C}}^{(i)} \widehat{\mathbf{G}}^{-1}(\widehat{\mathbf{w}}^T) + e^{sm} \\ &= (\mu \widehat{\mathbf{t}} \widehat{\mathbf{G}} + \mathbf{e}) \widehat{\mathbf{G}}^{-1}(\widehat{\mathbf{w}}^T) + e^{sm} \\ &= \mu \lceil q/2 \rceil + e' + e^{sm} \end{aligned}$$

where $e^{sm} = \sum_{i \in [N]} e_i^{sm}$ has norm $|e^{sm}| \le N B_{smdg}^{dec} = 2^{O(d\lambda \log \lambda)} B_\chi$ and $e' = \mathbf{e} \widehat{\mathbf{G}}^{-1}(\widehat{\mathbf{w}}^T)$ has norm $|e'| \le \beta'_{final} mN = 2^{O(d \log \lambda)} B_\chi$. Since $q = 2^{\omega(d\lambda \log \lambda)} B_\chi$ we have $|e' + e^{sm}| < q/4$ and correctness holds.

Simulatability: The simulator $\mathcal{S}^{thr}(\mu, \widehat{\mathbf{C}}, i, \{\mathbf{t}_j\}_{j\in[N]\setminus\{i\}})$, on input the secrets keys $\{\mathbf{t}_j\}_{j\neq i}$ the evaluated ciphertext $\widehat{\mathbf{C}} \in \mathbb{Z}_q^{nN\times mN}$ and the output value $\mu = \mathcal{C}(\mu_1, \ldots, \mu_\ell)$ encrypted in $\widehat{\mathbf{C}}$ outputs the *simulated partial decryption*:

$$p'_i = \mu\lceil q/2\rceil + e_i^{sm} - \sum_{i\neq j}\gamma_j \tag{1}$$

for $e_i^{sm} \xleftarrow{\$} [-B_{smdg}^{dec}, B_{smdg}^{dec}]$ where $\gamma_j = \mathbf{t}_j\widehat{\mathbf{C}}^{(j)}\widehat{\mathbf{G}}^{-1}(\widehat{\mathbf{w}}^T)$.

To see the indistinguishability note that, by the same calculation as used to argue correctness, we know that $\sum_{j\in[N]}\gamma_j = \mu\lceil q/2\rceil + e'$ where $|e'| \leq \beta'_{final}mN = 2^{O(d\log\lambda)}B_\chi$. Therefore if $p_i = \gamma_i + e_i^{sm}$ is the *real partial decryption* then

$$p_i = \mu\lceil q/2\rceil + e' + e_i^{sm} - \sum_{i\neq j}\gamma_j$$

The difference between the real value p_i and the simulated value p'_i is the noise e' of norm $|e'| = 2^{O(d\log\lambda)}B_\chi$. But by the smudging lemma 3.1, the distributions of e_i^{sm} and $e_i^{sm} + e'$ are statistically close since $e_i^{sm} \xleftarrow{\$} [-B_{smdg}^{dec}, -B_{smdg}^{dec}]$ where $B_{smdg}^{dec} = 2^{d\lambda\log\lambda}B_\chi$ so that $B_{smdg}^{dec}/|e'| \geq 2^\lambda$. Therefore the simulated partial decryption and the real one are statistically indistinguishable.

5.5 Bootstrapping

Note that the above MFHE scheme is leveled i.e., it depends on the multiplicative depth of the circuit to be computed. However, this dependency can be avoided easily by boot-strapping and assuming circular security. We briefly describe the straightforward procedure and omit the details.

During key generation, each party i chooses a key pair (sk_i, pk_i) and uses the MFHE scheme to encrypt the secret key sk_i under pk_i bit-by-bit.[7] It appends these encryptions to the public key. Later, given a sequence of public keys $\{pk_1, \ldots, pk_N\}$ anyone can create an expanded multi-key encryption of each sk_i using the MFHE expansion procedure. This allows us to use Gentry's bootstrapping technique [Gen09] to to "refresh" a highly noisy multi-key ciphertext by homomorphically computing the MFHE decryption procedure. Therefore, to compute a circuit of *arbitrary* depth, we only need to set the parameters of the MFHE scheme so as to be evaluate circuits of some *fixed* depth $d + 1$ where d is the depth of the MFHE decryption procedure.

Note that, by circular security it is assured that an encryption of a secret key under itself is semantically secure which implies that the semantic security of the above modified MFHE scheme remains intact.

6 Secure MPC via Threshold MFHE

Basic Template. We now present a protocol for general MPC, using any threshold multi-key fully homomorphic scheme. The protocol is based on the template discussed in the introduction which we recall below:

[7] We ignore the algebraic structure of the secret key here and assume each element in $\mathbb{Z}_q$ can be represented as a $\lceil\log(q)\rceil + 1$ binary string.

1. Each party individually chooses its own MFHE key pair (pk_i, sk_i), encrypts its input x_i under pk_i, and broadcasts the resulting ciphertext. At the end of this round, each party can homomorphically compute the desired function f on the received ciphertexts and derive a common multi-key ciphertext which encrypts the output $y = f(x_1, \ldots, x_N)$.
2. The parties run a distributed protocol for "threshold decryption" using their secret keys sk_i to decrypt the multi-key ciphertext and recover the output y in plaintext. In particular each party first generates partial decryptions p_i from the common (evaluated) ciphertext $\widehat{c}$ and then broadcasts them. Finally each party, on receiving all those partial decryptions can compute the final decryption y.

Our goal is to prove the security of this protocol (as least in the honest-but-curious setting, as a start). The natural attempt to construct a MPC simulator $\mathcal{S}$ would be to first use the simulator of threshold decryption, $\mathcal{S}^{thr}$ to replace the correct partial decryptions p_i with simulated ones p_i' and then use semantic security of the encryption to replace each ciphertext (broadcast in the first round) by encryptions of 0.

The Problem. Unfortunately, we notice that the simulatability of the threshold decryption does not suffice when there is more than one honest party. Essentially, our definition of simulation security for threshold decryption only allows us to simulate the partial decryption of a single party at a time while knowing the secret keys of all other parties. We cannot, however, simultaneously simulate the partial decryptions of (even) two honest parties without knowing either of their secret keys.

Solution. Essentially we solve the above problem by two steps. We first show that the "basic" protocol as described above is already secure when there is exactly one honest party. Then, later in Sec. 6.2 we extend the basic protocol to another protocol which can handle any arbitrary number of corruption. The extended protocol additionally requires only pseudorandom functions (PRFs) and thus no new assumptions are used. Combining, we get a protocol which securely realizes any functionality against any arbitrary number of corruptions. Below we provide the basic protocol from any MFHE scheme and prove security against exactly $N-1$ corruptions. Later in Sec. 6.2 we present the extension in detail.

Semi-Malicious Security. Following [AJLA+12], we will actually prove that the above protocol satisfies something called "semi-malicious" security which is stronger than honest-but-curious. Intuitively, it means that adversarial parties need to follow the protocol specification, but can use arbitrary values for their random coins. In fact, the adversary only needs to decide on the input and the random coins to use for each party in each round at the time that the party sends the first message. See Appendix A for a formal definition. We will then rely on a theorem of [AJLA+12] showing that one can compile any such protocol which is secure in the semi-malicious setting into one that is secure in the fully malicious setting, without adding any rounds, by using non-interactive zero-knowledge proofs (NIZKs).

6.1 Protocol Secure against Exactly $N-1$ Corruptions

Let $f : (\{0,1\}^{\ell_{in}})^N \to \{0,1\}^{\ell_{out}}$ be the function to compute. Let d be the depth of the circuit for f.

Preprocessing. Run $\mathsf{setup} \leftarrow \mathsf{MFHE.Setup}(1^\lambda, 1^d)$. All the parties share the common setup.

Input: Each party P_k has input $\mathbf{x}_k \in \{0,1\}^{\ell_{in}}$.

<u>The Protocol:</u>

Round I. Each party P_k executes the following steps.

- Generate a key-pair $(sk_k, pk_k) \leftarrow \mathsf{MFHE.Keygen}(\mathsf{setup})$.
- Encrypt the message bit-by-bit:

$$\{c_{k,j} \leftarrow \mathsf{MFHE.Encrypt}(pk_k, \mathbf{x}_k[j])\}_{j\in[\ell_{in}]}.$$

- Broadcast the public-key and the ciphertexts $(pk_k, \{c_{k,j}\}_{j\in[\ell_{in}]})$.

Round II. Each party P_k on receiving values $\{pk_i, c_{i,j}\}_{i\in[N]\setminus\{k\}, j\in[\ell_{in}]}$ executes the following steps:

- First expand each $c_{i,j}$:

$$\{\widehat{c}_{i,j} \leftarrow \mathsf{MFHE.Expand}((pk_1, \ldots, pk_N), i, c_{i,j})\}_{i\in[N], j\in[\ell_{in}]}$$

- Run the evaluation algorithm to generate the evaluated ciphertext:

$$\{\widehat{c}_j \leftarrow \mathsf{MFHE.Eval}(f_j, (\widehat{c}_{1,1}, \ldots, \widehat{c}_{N,\ell_{in}}))\}_{j\in[\ell_{out}]}.$$

  where f_j is the boolean function for j-th bit of the output of f.
- Finally all the parties concurrently take part in one-round threshold decryption to obtain the output message bit-by-bit as follows:
  - Each P_k computes the partial decryption for all $j \in [\ell_{out}]$:

$$p_k^{(j)} \leftarrow \mathsf{MFHE.PartDec}(\widehat{c}_j, (pk_1, \ldots, pk_N), k, sk_k)$$

  - P_k broadcasts all the values $\{p_k^{(j)}\}_{j\in\ell_{out}}$.

Output. On receiving all the values $\{p_i^{(j)}\}_{i\in[N], j\in[\ell_{out}]}$ run the final decryption to obtain the j-th output bit: $\{y_j \leftarrow \mathsf{MFHE.FinDec}(p_1^{(j)}, \ldots, p_N^{(j)})\}_{j\in[\ell_{out}]}$. Output $y = y_1 \ldots y_{\ell_{out}}$.

Figure 2: π_f: A basic MPC protocol for f secure against $N-1$ corruptions

The protocol, given in Figure 2, realizes general multiparty computation for any polynomial-time deterministic functions f which produces a common output for all parties. It does so with respect to a static *semi-malicious attackers* corrupting exactly $N-1$ parties (see Appendix A). Formally we prove the following theorem.

Theorem 6.1. *Let f be a poly-time computable deterministic function with N inputs and 1 output. Let the scheme* $\mathsf{MFHE} = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Encrypt}, \mathsf{Expand}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$ *be a multi-key FHE scheme with threshold decryption. Then the protocol π_f described in Fig. 2 UC-realizes the function f against any static semi-malicious adversary corrupting exactly $N-1$ parties.*

Proof: The correctness of the protocol follows in a straightforward way from the correctness of the underlying threshold MFHE scheme.

To prove security basically we need to construct an efficient (PPT) simulator $\mathcal{S}$ for any adversary corrupting exactly $N-1$ parties. Let $\mathcal{A}$ be a static semi-malicious adversary and P_h be the *only* honest party. The simulator simulates the protocol execution on behalf of the honest party P_h as follows.

The Simulator. In round-I, the simulator encrypts 0s instead of the real input bits of the honest party P_h. After round-I it gets the inputs and the secret keys of the $N-1$ corrupt parties from the "witness tape". It gives these inputs to the ideal functionality and receives the output bits y_j for each $j \in [\ell_{out}]$. At this point it can also compute the evaluated ciphertexts $\widehat{c}_j$. Then it computes the simulated partial decryptions for the honest party $\widetilde{p}_h^{(j)} \leftarrow \mathcal{S}^{thr}(y_j, \widehat{c}_j, h, \{sk_i\}_{i\in[N]\setminus\{h\}})$ and broadcast those in round-II instead of correctly computed partial decryptions $p_h^{(j)}$ generated via $\mathsf{MFHE.PartDec}(\cdots)$.

Hybrid Games. We now define a series of *hybrid games* that will be used to prove the indistinguishability of the real and ideal worlds:

$$\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \overset{\text{comp}}{\approx} \mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}} \tag{2}$$

The output of each game is always just the output of the environment.

The game $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$**:** This is exactly an execution of the protocol π in the real world with environment $\mathcal{Z}$ and semi-malicious adversary $\mathcal{A}$.

The game $\mathrm{HYB}^1_{\pi,\mathcal{A},\mathcal{Z}}$**:** In this game, we modify the real world experiment as follows. Assume (as a mental experiment) that P_h is given the all the secret keys $\{sk_i\}_{i\in[N]\setminus\{h\}}$ (as written on the "witness tape" of the adversary) after round I. In the second round, instead of broadcasting a correctly generated partial decryptions $p_h^{(j)}$ generated via $\mathsf{MFHE.PartDec}(\cdots)$, it broadcasts simulated ones $\{\widetilde{p}_h^{(j)} \leftarrow \mathcal{S}^{thr}(y_j, \widehat{c}_j, h_1, \{sk_i\}_{i\neq h})\}_{j\in[\ell_{out}]}$.

The game $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$**:** This is similar to the game $\mathrm{HYB}^1_{\pi,\mathcal{A},\mathcal{Z}}$ except instead of encrypting its real input, P_h now broadcasts encryption of 0s in the first round.

Claim 6.2. $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}} \overset{\text{stat}}{\approx} \mathrm{HYB}^1_{\pi,\mathcal{A},\mathcal{Z}}$

Proof: Notice that, the only change between those experiments are that, the partial decryption of party P_h is generated through simulator $\mathcal{S}^{thr}$ instead of correctly using $\mathsf{MFHE.PartDec}$. By simulatability of threshold decryption the partial decryptions are statistically indistinguishable hence so are the experiments. ■

Claim 6.3. $\mathrm{HYB}^1_{\pi,\mathcal{A},\mathcal{Z}} \overset{\text{comp}}{\approx} \mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$

Proof: The only change between those experiments are in generating encryptions of party P_h. By semantic security of the underlying MFHE the encryptions are computationally indistinguishable. Hence the experiments are also computationally indistinguishable. Note that here it is possible to use the semantic security as the partial decryptions of P_h in both the experiments are simulated and hence independent of the secret key sk_h. ■

This concludes the proof of the theorem. ■

6.2 An Extended Protocol for Arbitrary Many Corruptions

In this section we construct an "extended" MPC protocol $\widehat{\pi}_f$ which securely computes any function f against any semi-malicious adversary that can corrupt any $t \in [N]$ parties. We do so by relying on the "basic" MPC protocol π_f from the previous section, which is secure against a semi-malicious adversary that corrupts exactly $N-1$ parties. To compute a function f, our extedned protocol simply runs the basic protocol $\pi_{\widehat{f}}$ on an extended function $\widehat{f}$ defined as follows.

Definition 6.4. *(Extended function) For any polynomial $\ell_{in}, \ell_{out}, N \in \mathbb{N}$ let $f : \{\{0,1\}^{\ell_{in}}\}^N \to \{0,1\}^{\ell_{out}}$ be a poly-time computable function and $\mathsf{PRF} : \{0,1\}^\lambda \times [N] \to \{0,1\}^{\ell_{in}}$ be a PRF. Then we define an extended function $\widehat{f} : \{\{0,1\}^{\ell_{in}} \times \{1,2,3\} \times \{0,1\}^\lambda\}^N \to \{0,1\}^{\ell_{out}}$ which takes as input $((\mathbf{x}_1, \mathtt{mode}_1, \mathbf{z}_1), \ldots, (\mathbf{x}_N, \mathtt{mode}_N, \mathbf{z}_N))$ and does the following:*

- *If $\forall\ i \in [N]$, $\mathtt{mode}_i = 1$ then output $f(\mathbf{x}_1, \ldots, \mathbf{x}_N)$.*
- *If $\exists$ unique $i \in [N]$ such that $\mathtt{mode}_i = 2$ then let $K := \mathbf{z}_i$. For all $j \in [N]$:*
  - *If $\mathtt{mode}_j = 3$ then set $\mathbf{x}'_j := \mathsf{PRF}(K, j) \oplus \mathbf{x}_j$.*
  - *Else set $\mathbf{x}'_j := \mathbf{x}_j$.*

  Output $f(\mathbf{x}'_1, \ldots, \mathbf{x}'_N)$.
- *Otherwise output $0^{\ell_{out}}$.*

Roughly speaking, the extended function does the same thing as the original function if all the inputs have $\mathtt{mode}_i = 1$. However, if there is one special party with $\mathtt{mode}_i = 2$ then the function uses a PRF key $K = \mathbf{z}_i$ provided by that party to "decrypt" the inputs of all the parties with $\mathtt{mode}_j = 3$.

We define an "extended protocol" $\widehat{\pi}_f$ in Figure 3. It essentially just runs the original basic protocol $\pi_{\widehat{f}}$ with an extended function $\widehat{f}$ and appropriately extended inputs.

We now prove that that the extended protocol $\widehat{\pi}_f$ is secure against any arbitrary number of semi-malicious corruptions.

Theorem 6.5. *Let f be a function with N inputs and 1 output. Let $\mathsf{PRF} : \{0,1\}^\lambda \times [N] \to \{0,1\}^{\ell_{out}}$ be a PRF. Then, under the LWE assumption, the protocol $\widehat{\pi}_f$ shown in Figure 3 UC-realizes f against a static semi-malicious adversary that can corrupt any number of parties.*

Let $f : \{\{0,1\}^{\ell_{in}}\}^N \to \ell_{out}$ be the function we wish to compute and let and $\mathsf{PRF} : \{0,1\}^\lambda \times [N] \to \{0,1\}^{\ell_{in}}$ be a PRF. Let $\widehat{f} : \{\{0,1\}^{\ell_{in}} \times \{1,2,3\} \times \{0,1\}^\lambda\}^N \to \{0,1\}^{\ell_{out}}$ be the corresponding extended function (Definition 6.4). Let $\pi_{\widehat{f}}$ be the protocol from Figure 2 applied to the extended function $\widehat{f}$. The extended protocol $\widehat{\pi}_f$ is defined as follows:

Setup: The setup is the same as the that of the protocol $\pi_{\widehat{f}}$.

Input: Each party P_k has input $\mathbf{x}_k \in \{0,1\}^{\ell_{in}}$. Additionally each party sets $\mathtt{mode}_k := 1$, $\mathbf{z}_k := \mathsf{0}$, and defines its extended input as $\widehat{\mathbf{x}}_k := (\mathbf{x}_k, \mathtt{mode}_k, \mathbf{z}_k) \in \{0,1\}^{\widehat{\ell_{in}}}$ where $\widehat{\ell_{in}} = \ell_{in} + \lambda + 2$. [a]

Protocol: The parties run the protocol $\pi_{\widehat{f}}$ using the extended inputs $\{\widehat{\mathbf{x}}_k\}_{k \in [N]}$. They output whatever $\pi_{\widehat{f}}$ outputs.

[a] Here any $\mathsf{0}$ denotes a string of 0s of appropriate size. We abuse notation for simplification.

Figure 3: $\widehat{\pi}_f$: Extended protocol secure against any any number of semi-malicious corruptions.

Proof: Correctness is clear from the definition of extended function. In particular note that each extended input has the flag $\mathtt{mode}_k$ set to 1 which means the extended function computes the correct output $f(\mathbf{x}_1, \ldots, \mathbf{x}_N)$.

Let $\mathcal{A}$ be a static semi-malicious adversary against $\widehat{\pi}_f$, and let I be the set of corrupt parties with $|I| = t$. We can assume $0 < t \leq N - 1$ as otherwise security is trivial to show. Let $(h_1, h_2, \ldots, h_{N-t}) \in [N] \setminus I$ be some arbitrary ordering of the honest parties $\{P_{h_i}\}_{i \in [N-t]}$.

The Simulator $\mathcal{S}$. The simulator $\mathcal{S}$ executes the protocol with the adversary $\mathcal{A}$ acting on behalf oh the honest parties as specified, except that:

- In round-I: For party P_{h_1} the simulator uses the input $\widehat{\mathbf{x}}_{h_1} = 0^{\widehat{\ell_{in}}}$ and for all other honest parties P_{h_i}, $i > 1$ it uses the input $\widehat{\mathbf{x}}_{h_i} = (\mathbf{x}'_{h_i}, \mathtt{mode}_{h_i} = 3, \mathbf{z}_{h_i} = 0^{\ell_{in}})$ where $\mathbf{x}'_{h_i} \leftarrow \{0,1\}^{\ell_{in}}$ is chosen uniformly at random. It follows round I of the protocol $\pi_{\widehat{f}}$ honestly with these inputs.
- After round-I it gets the inputs and the secret keys of the t corrupt parties from the "witness tape". It gives the inputs to the ideal functionality and received the output $y = (y_1, \ldots, y_{\ell_{out}})$.
- In round-II: The simulator homomorphically computes the evaluated ciphertexts $\widehat{c}_j$ for $j \in \ell_{out}$, encrypting the output. For honest parties P_{h_i}, $i > 1$ it gives the correctly generated partial decrpytions $p^{(j)}_{h_i} \leftarrow \mathsf{MFHE.PartDec}(\widehat{c}_j, (pk_1, \ldots, pk_N), h_i, sk_{h_i})$ using the secret key sk_{h_i} chosen for that party in round I. For the honest party P_{h_1} it computes simulated partial decryptions $\widetilde{p}^{(j)}_{h_1} \leftarrow \mathcal{S}^{thr}(y_j, \widehat{c}_j, h_1, \{sk_i\}_{i \in [N] \setminus \{h_1\}})$ using the secret keys sk_i of the other honest and corrupted parties.

Below we argue that the simulated view is computationally indistinguishable from the view of the adversary in the real world:

$$\mathrm{IDEAL}_{\mathcal{F},\widetilde{\mathcal{S}},\mathcal{Z}} \overset{\text{comp}}{\approx} \mathrm{REAL}_{\widehat{\pi},\mathcal{A},\mathcal{Z}} \tag{3}$$

Hybrid Games. We move from the real execution to the ideal execution through a series of hybrid games as defined below. The output of each game is always just the output of the environment.

The game $\text{REAL}_{\widehat{\pi},\mathcal{A},\mathcal{Z}}$**:** This is exactly an execution of the protocol $\widehat{\pi}_f$ in the real world with environment $\mathcal{Z}$ and semi-malicious adversary $\mathcal{A}$. Recall that, here each honest party P_k follows protocol $\pi_{\widehat{f}}$ with inputs $(\mathbf{x}_k, 1, 0)$.

The game $\text{HYB}^{1,1}$**:** In this game, we modify the real world experiment as follows. After round I, let $\{sk_i\}_{i\in[N]}$ be the secret keys chosen by the honest parties and written on the "witness tape" of the corrupt parties. Let $y = (y_1, \ldots, y_{\ell_{out}})$ be the output of the ideal functionality f when given the actual inputs of the honest parties and the inputs written on the "witness tape" of the corrupt parties. In the second round of the protocol, instead of party P_{h_1} broadcasting correctly generated partial decryptions $p_{h_1}^{(j)}$ generated via $\mathsf{MFHE.PartDec}(\cdots)$, it broadcasts simulated partial decryptions $\{\widetilde{p}_h^{(j)} \leftarrow \mathcal{S}^{thr}(y_j, \widehat{c}_j, h_1, \{sk_i\}_{i\neq h_1})\}_{j\in[\ell_{out}]}$.

The game $\text{HYB}^{1,2}$**:** In this same as $\text{HYB}^{1,1}$ except that, in round I, instead of P_{h_1} using the value $(\mathbf{x}_{h_1}, \mathtt{mode}_{h_1} = 1, \mathbf{z}_{h_1} = 0)$ as its input for $\pi_{\widehat{f}}$, it now chooses a random PRF key $K \leftarrow \{0,1\}^{\lambda}$ and uses the value $(\mathbf{x}_{h_1}, \mathtt{mode}_{h_1} = 2, \mathbf{z}_{h_1} = K)$ as its input.

The game $\text{HYB}^{1,3}$**:** In this same as $\text{HYB}^{1,2}$ except that, in round II, the party P_{h_1} goes back to broadcasting the correctly generated partial decryptions $p_{h_1}^{(j)}$ generated via $\mathsf{MFHE.PartDec}(\cdots)$.

The games $\{\text{HYB}^{i,1}\}_{i=2,\ldots,N-t}$**:** Each game $\text{HYB}^{i,1}$ is same as the game $\text{HYB}^{i-1,3}$ except that, in round II of the protocol, instead of party P_{h_i} broadcasting correctly generated partial decryptions $p_{h_i}^{(j)}$ generated via $\mathsf{MFHE.PartDec}(\cdots)$, it broadcasts simulated ones $\{\widetilde{p}_{h_i}^{(j)} \leftarrow \mathcal{S}^{thr}(y_j, \widehat{c}_j, h_i, \{sk_k\}_{k\neq h_i})\}_{j\in[\ell_{out}]}$, where sk_k and y are defined the same way as in $\text{HYB}^{1,1}$.

The games $\{\text{HYB}^{i,2}\}_{i=2,\ldots,N-t}$**:** Each game $\text{HYB}^{i,2}$ is same as the game $\text{HYB}^{i,1}$ except that, in round I of the protocol, instead of P_{h_i} using the value $(\mathbf{x}_{h_i}, 1, 0)$ as its input for $\pi_{\widehat{f}}$, it now uses the input $(\mathbf{x}_{h_i} \oplus \mathsf{PRF}(K, h_i), 3, 0)$, where K is the same PRF key that was defined in Hybrid $\text{HYB}^{1,2}$.

The games $\{\text{HYB}^{i,3}\}_{i=2,\ldots,N-t}$**:** Each game $\text{HYB}^{i,3}$ is same as the game $\text{HYB}^{i,2}$ except that, in round II of the protocol, the party P_{h_i} goes back to broadcasting the correctly generated partial decryptions $p_{h_i}^{(j)}$ generated via $\mathsf{MFHE.PartDec}(\cdots)$.

The game $\text{HYB}^{N-t+1,1}$**.** This is the same as $\text{HYB}^{N-t,3}$ except that, in round II of the protocol, instead of party P_{h_1} broadcasting correctly generated partial decryptions $p_{h_1}^{(j)}$ generated via $\mathsf{MFHE.PartDec}(\cdots)$, it broadcasts simulated ones $\{\widetilde{p}_{h_i}^{(j)} \leftarrow \mathcal{S}^{thr}(y_j, \widehat{c}_j, h_i, \{sk_k\}_{k\neq h_i})\}_{j\in[\ell_{out}]}$, where sk_k and y are defined the same way as in $\text{HYB}^{1,1}$.

The game $\text{HYB}^{N-t+1,2}$**.** This is the same as $\text{HYB}^{N-t+1,1}$ except that, in round I of the protocol, instead of P_{h_1} using the value $(\mathbf{x}_{h_1}, 2, K)$ as its input to $\pi_{\widehat{f}}$, it now uses the value $0^{\widehat{\ell_{in}}}$.

The game $\text{IDEAL}_{\mathcal{F},\widetilde{\mathcal{S}},\mathcal{Z}}$**:** This is the ideal-world execution with simulator $\mathcal{S}$ and environment $\mathcal{Z}$. It is the same as $\text{HYB}^{N-t+1,2}$ except that, instead of the honest parties P_{h_i}, $i > 1$ using the values $(\mathbf{x}_{h_i} \oplus \mathsf{PRF}(K, h_i), 3, 0)$ as their inputs to $\pi_{\widehat{f}}$ they use the inputs $(\mathbf{x}'_{h_i}, 3, 0)$ where $\mathbf{x}'_h$ is random.

For notational convenience let us rename the real game $\text{REAL}_{\widehat{\pi},\mathcal{A},\mathcal{Z}}$ as $\text{HYB}^{0,3}$. Let us also define $h_{N-t+1} = h_1$.

Claim 6.6. *For all $i \in [N-t+1]$: $\text{HYB}^{i-1,3} \overset{\text{stat}}{\approx} \text{HYB}^{i,1}$.*

Proof: Notice that, the only change between these hybrids is in whether a single party P_{h_i} computes its partial decryptions honestly via $p^{(j)}_{h_i} \leftarrow \mathsf{MFHE.PartDec}(\widehat{c}_j, (pk_1, \ldots, pk_N), h_i, sk_{h_i})$ or whether it gives simulated ones $\widetilde{p}^{(j)}_{h_i} \leftarrow \mathcal{S}^{thr}(y_j, \widehat{c}_j, h_i, \{sk_k\}_{k \neq h_i})$. Each of the evaluated ciphertexts $\widehat{c}_j$ is an encryption of y_j by correctness and the way we chose the inputs of all honest parties. Therefore the above indistinguishability holds by the simulatability of partial decryption of the MFHE scheme. ■

Claim 6.7. *For all $i \in [N-t+1]$: $\text{HYB}^{i,1} \overset{\text{comp}}{\approx} \text{HYB}^{i,2}$.*

Proof: Notice that, the only change between these hybrids is in the value encrypted by the party P_{h_i} in round I. However, the secret key sk_{h_i} of that party is not used anywhere in either of these hybrids. Therefore the hybrids are indistinguishable by the semantic security of the MFHE scheme. ■

Claim 6.8. *For all $i \in [N-t]$: $\text{HYB}^{i,2} \overset{\text{stat}}{\approx} \text{HYB}^{i,3}$.*

Proof: Notice that, the only change between these hybrids is in whether a single party P_{h_i} computes its partial decryptions honestly via $p^{(j)}_{h_i} \leftarrow \mathsf{MFHE.PartDec}(\widehat{c}_j, (pk_1, \ldots, pk_N), h_i, sk_{h_i})$ or whether it gives simulated ones $\widetilde{p}^{(j)}_{h_i} \leftarrow \mathcal{S}^{thr}(y_j, \widehat{c}_j, h_i, \{sk_k\}_{k \neq h_i})$. Each of the evaluated ciphertexts $\widehat{c}_j$ is an encryption of y_j by correctness and the way we chose the inputs of all honest parties. Therefore the above indistinguishability holds by the simulatability of partial decryption of the MFHE scheme. ■

Claim 6.9. $\text{HYB}^{N-t+1,2} \overset{\text{comp}}{\approx} \text{IDEAL}_{\mathcal{F},\widetilde{\mathcal{S}},\mathcal{Z}}$

Proof: The only difference between these hybrids is whether the inputs of the honest parties P_{h_i} $i \in [2, \ldots, N-t]$ are of the form $(\mathbf{x}_{h_i} \oplus \mathsf{PRF}(K, h_i), 3, 0^{\ell_{in}})$ or $(\mathbf{x}'_{h_i}, 3, 0^{\ell_{in}})$ where $\mathbf{x}'_{h_i} \leftarrow \{0,1\}^{\ell_{in}}$. This is indistinguishable by PRF security. Note that the key K is not used anywhere else in either of these hybrids other than the PRF evaluations $\mathsf{PRF}(K, h_i)$. ■

Combining the above claims, we get $\text{IDEAL}_{\mathcal{F},\widetilde{\mathcal{S}},\mathcal{Z}} \overset{\text{comp}}{\approx} \text{REAL}_{\widehat{\pi},\mathcal{A},\mathcal{Z}}$ which concludes the proof of Theorem 6.5. ■

6.3 Extensions and Applications

Generalized functionalities. Our protocol (Fig. 2) considers deterministic functionalities where all the parties receive the same output. One can extend that to handle randomized functionalities and individual output in a straightforward manner using known standard techniques just like [AJW11]. We refer to [AJW11] for more details.

Fully malicious adversary. Our protocol protects only against semi-malicious adversaries. However, since we are in the CRS model such protocol can be generically converted to one secure against fully-malicious adversary using non-interactive zero-knowledge (NIZK) arguments. For more detail on this again we refer to [AJW11].

Communication Complexity. Although our main focus was on round complexity, we mention that our scheme also achieves essentially optimal communication complexity which is only proportional to the total input size, output size and circuit depth. We can get rid of the reliance on circuit depth by using bootstrapping and relying on circular security: each party would simply send a GSW encryption of its secret key under its public key and then we would perform a boostrapping step after each homomorphic operation to reduce the noise in the ciphertext.

Computation on the Web. Our results also relate to the idea of "computation on the web" [HLP11] where parties can't interact with each other but can only interact with some central website without further coordination. Using our scheme (or any 2 round protocol) each party needs to log in twice: once to give its ciphertext to the sever and once to give a partial decryption of the output.

7 Conclusions

We have shown how to implement MPC with only two rounds of interaction by relying on the LWE assumption (and NIZKs for malicious security). Several interesting open problems remain. Firstly, is is possible to get a 2 round MPC protocol under general assumptions such as the existence of oblivious transfer? Secondly, is it possible to get a protocol that achieves adaptive security? A recent work of [GP15] does this using indistinguishability obfuscation (iO) but it remains an open problem to do this using more standard assumptions such as LWE. Lastly, it would be interesting to get a 2 round protocol in the honest-but-curious model without a CRS. One way to achieve this would be to a build a threshold multi-key FHE without any common public parameters.

8 Acknowledgments

We thank Antigoni Polychroniadou for her helpful comments and helping us find and fix a subtle error with the extended MPC protocol given in a previous version of this work.

References

[AIK05] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. In *IEEE Conference on Computational Complexity*, pages 260–274, 2005.

[AJLA$^+$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 *Advances in Cryptology–EUROCRYPT 2012*, pages 483–501. Springer Berlin Heidelberg, 2012.

[AJW11] Gilad Asharov, Abhishek Jain, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. Cryptology ePrint Archive, Report 2011/613, 2011. http://eprint.iacr.org/.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 297–314. Springer, 2014.

[BD10] Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In *TCC*, 2010.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT*, pages 169–188, 2011.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.

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

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *FOCS*, 2011.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT*, 2001.

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 630–656, 2015.

[DN03] Ivan Damgård and Jesper Buus Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *CRYPTO*, pages 247–264, 2003.

[FH96] Matthew K. Franklin and Stuart Haber. Joint encryption and message-efficient secure computation. *J. Cryptology*, 9(4):217–232, 1996.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA,* *February 24-26, 2014. Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 74–94. Springer, 2014.

[GMPP15] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. Manuscript, via personal communication, October 2015.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In *STOC*, pages 695–704, 2011.

[GP15] Sanjam Garg and Antigoni Polychroniadou. Two-round adaptively secure MPC from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 614–637. Springer, 2015.

[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, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

[HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *CRYPTO*, pages 132–150, 2011.

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

[JJ00] Markus Jakobsson and Ari Juels. Mix and match: Secure function evaluation via ciphertexts. In *ASIACRYPT*, pages 162–177, 2000.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *CRYPTO*, pages 335–354, 2004.

[KOS03] Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Round efficiency of multi-party computation with a dishonest majority. In *EUROCRYPT*, pages 578–595, 2003.

[Lin01] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. In *CRYPTO*, pages 171–189, 2001.

[LP11] Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In *STOC*, pages 705–714, 2011.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 1219–1234. ACM, 2012.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, 2012.

[MSS11] Steven Myers, Mona Sergi, and Abhi Shelat. Threshold fully homomorphic encryption and secure computation. In *eprint 2011/454*, 2011.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *STOC*, pages 333–342, 2009.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.

A Definitions of MPC

The content of this section is taken essentially verbatim from [AJW11] and is included for completeness.

A.1 The Universal Composability Framework (UC)

We work in the standard universal composability framework of Canetti [Can01] with static corruption. The UC framework defines a PPT environment $\mathcal{Z}$ that is invoked on security parameter 1^λ and an auxiliary input z, and oversees the execution of a protocol in one of two worlds. The "ideal world" executions involves dummy parties $\tilde{P}_1, \ldots, \tilde{P}_N$, an ideal adversary $\mathcal{S}$ who may corrupt some of the dummy parties, and a functionality $\mathcal{F}$. The "real world" execution involves the PPT parties $P_1, \ldots, P_N$, and a real-world adversary $\mathcal{A}$ who may corrupt some of the parties. The environment $\mathcal{Z}$ chooses the inputs of the parties, may interact with the ideal / real adversary during the execution, and at the end of the execution need to decide whether a real or ideal execution has been taken place. The output of the execution is simply the output of the environment. We refer to [Can01] for further details.

Let $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(1^\lambda, z)$ denote the random variable describing the output of the environment $\mathcal{Z}$ after interacting in the ideal process with adversary $\mathcal{S}$, the functionality $\mathcal{F}$, on security parameter 1^λ and input z. Let $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ denote the ensemble $\{\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(1^\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}$. Similarly, let $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(1^\lambda, z)$ denote the random variable describing the output of the environment $\mathcal{Z}$ after interacting with the adversary $\mathcal{A}$ and parties running protocol π on security parameter λ, and input z. Let $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ denote the ensemble $\{\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(1^\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}$.

Definition A.1. *For $N \in \mathbb{N}$, let $\mathcal{F}$ be an N-ary functionality, and let π be a N-party protocol. We say that π* securely realizes *$\mathcal{F}$ if for any* PPT *adversary $\mathcal{A}$ there exists a* PPT *ideal adversary $\mathcal{S}$ such that for any* PPT *environment $\mathcal{Z}$ we have:*

$$\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \overset{\text{comp}}{\approx} \text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$$

We sometime want to restrict the definition so that it quantifies over adversaries from a certain class: semi-honest adversaries, or malicious adversary that corrupted only a certain number of parties. That is done in a straightforward way.

General functionality. We consider the general UC–functionality $\mathcal{F}$, which securely evaluates any function $f : (\{0,1\}^{\ell_{in}})^N \to (\{0,1\}^{\ell_{out}})^N$. The functionality $\mathcal{F}_f$ is parameterized with a function f and is defined as follows:

- Each party P_i sends (x_i, sid) to the functionality.
- Once all parties send their inputs, evaluate $(y_1, \ldots, y_N) \leftarrow f(x_1, \ldots, x_N)$.
- Send to each party P_i the output (y_i, sid).

A.2 Security Against Semi-Malicious Adversaries

As a stepping stone towards realizing the standard definition of secure multi-party computation against active adversaries, we provide definition of a *semi-malicious* adversary below which is taken verbatim from [AJW11]. We formalize security against semi-malicious adversaries, however one can use the generic transformation provided in [AJW11] from an MPC protocol that is secure against semi-malicious adversaries, to a protocol that is secure against fully malicious adversaries.

Semi–malicious adversary. A *semi-malicious adversary* is modeled as an interactive Turing machine (ITM) which, in addition to the standard tapes, has a special *witness tape*. In each round of the protocol, whenever the adversary produces a new protocol message m on behalf of some party P_k, it must also write to its special witness tape *some* pair (x, r) of input x and randomness r that *explains its behavior*. More specifically, all of the protocol messages sent by the adversary on behalf of P_k up to that point, including the new message m, must exactly match the honest protocol specification for P_k when executed with input x and randomness r. Note that the witnesses given in different rounds need not be consistent. Also, we assume that the attacker is rushing and hence may choose the message m and the witness (x, r) in each round adaptively, after seing the protocol messages of the honest parties in that round (and all prior rounds). Lastly, the adversary may also choose to abort the execution on behalf of P_k in any step of the interaction.

Definition A.2. *We say that a protocol π* securely realizes $\mathcal{F}$ for semi-malicious adversaries *if it satisfies Definition A.1 when we only quantify over all semi-malicious adversaries $\mathcal{A}$.*

We remark that this definition captures the semi-honest adversary who always follows the protocol with honestly chosen random coins (and can be easily modified to write those on its witness tape). On the other hand, a semi-malicious adversary is more restrictive than a fully malicious adversary, since its behavior follows the protocol with *some* input and randomness which it must *know*. Note that the semi-malicious adversary may choose the different input or random tape in an adaptive fashion using any PPT strategy according to the partial view it has seen.