# Multikey FHE in the Plain Model

Prabhanjan Ananth[1], Abhishek Jain[2], Zhengzhong Jin[2], and Giulio Malavolta[3]

[1]University of California Santa Barbara
[2]Johns Hopkins University
[3]UC Berkeley & Carnegie Mellon University

**Abstract**

We construct a multikey fully-homomorphic encryption scheme (multikey FHE) with one-round threshold decryption in the plain model, i.e. without a trusted setup, assuming the intractability of standard problems over lattices. Prior to our work, such a scheme was only known to exist assuming sub-exponentially hard indistinguishability obfuscation.

## 1 Introduction

Fully-homomorphic encryption [Gen09] (FHE) allows one to compute arbitrary functions on encrypted data. An important limitation of FHE is that it requires all of the data to be encrypted under the same public key in order to perform homomorphic evaluations. Consequently, López-Alt et al. [LTV12] proposed a multiparty extension of FHE, namely, *multikey FHE*, where each party can sample a key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ locally and encrypt its message under its own public key. Then one can publicly evaluate any (polynomially computable) function over the resulting ciphertexts $c_i = \mathsf{Enc}(\mathsf{pk}_i, m_i)$, each encrypted under an independently sampled public key. Naturally, decrypting the resulting multikey ciphertext requires one to know *all* the secret keys for the parties involved.

In this work we are interested in multikey FHE schemes with a 1 round decryption protocol [MW16]: Given a multikey ciphertext $c = \mathsf{Enc}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), f(m_1, \ldots, m_N))$, the decryption consists of (i) a local phase, where each parties independently computes a decryption share $\rho_i$ using its secret key $\mathsf{sk}_i$, and a (ii) public phase, where the plaintext $m$ can be publicly recovered from the decryption shares $(\rho_1, \ldots, \rho_N)$.

Other than being an interesting primitive on its own, multikey FHE with 1 round decryption implies a natural solution for secure multi-party computation (MPC) with *optimal* round complexity and communication complexity independent of the size of the function being computed [MW16]. Additionally, multikey FHE with 1 round decryption has interesting applications to functional encryption [AJS17], program obfuscation [AJN+16], time-lock puzzles [MT19, BDGM19], and many other cryptographic primitives. To the best of our knowledge, all known multikey FHE schemes with 1 round decryption assume a trusted setup [CM15, MW16, BP16, PS16] or require non-standard assumptions, such as the existence of sub-exponentially secure general-purpose obfuscation [DHRW16].

### 1.1 Our Results

In this work we build a multikey FHE with 1 round decryption in the plain model, i.e. without a trusted setup, from standard assumptions over lattices. This resolves a central open question

in the area, which was explicitly stated in [MW16, BP16, CCS19]. Specifically, we prove the following theorem.

**Theorem 1.1** (Informal)**.** *If the Learning with errors (LWE), Ring LWE and the Decisional Small Polynomial Ratio (DSPR) problems*[1] *are hard, then there exists leveled multikey FHE. Additionally assuming circular security of our scheme, there exists multikey FHE.*

The RingLWE and the DSPR problems are needed to prove semantic security of the multikey FHE (without 1 round decryption) proposed in [LTV12]. Stated differently, the semantic security of such a scheme implies the existence of a multikey FHE with 1 round decryption. Our construction achieves a relaxed security notion where, among other differences, we require computational indistinguishability of simulated decryption shares, whereas the works of [MW16, BP16, PS16] achieved statistical indistinguishability. (See Section 2.4 for details.) To the best of our knowledge, this definition suffices for all applications of multikey FHE.

Our work builds on the recent result of Ananth et al. [AJJ20] who study a related notion of multiparty homomorphic encryption (MHE), which is defined similarly to multikey FHE, with the key difference that the computation required to reconstruct the output is at least as high as recomputing the whole function. Assuming the hardness of LWE, Ananth et al. provide a construction of MHE with semi-honest security, where the size of decryption shares depends on the depth of the circuit being evaluated. Our work resolves all of these limitations of their work; in particular, we construct a multikey FHE scheme with semi-malicious security (as in [MW16]) with compact decryption and without depth-dependence (albeit with a circular-security assumption in the latter case, as in prior works).

As an important corollary of our main result, we obtain a 2-round MPC protocol for general functions from the same assumptions, with communication complexity *independent* of the circuit size. This is in contrast with prior 2-round protocols in the plain model [QWW18, ABJ$^+$19, AJJ20] where the communication complexity depends on the *depth* of the circuit being evaluated. Since 2 rounds are optimal, this settles the question of the communication complexity of multiparty computation (from standard assumptions) up to polynomial factors in the security parameter.

## 1.2 Technical Overview

In the following we give a high-level overview of the techniques that we develop in order to achieve our result and we refer the reader to the technical sections for precise statements.

**Multikey FHE from MPC.** Up until now, multikey FHE (with 1 round decryption) has been thought of as a tool to construct 2-round MPC protocols with low communication. However, we seem to have reached a roadblock in the kind of multikey FHE scheme that we know how to construct from standard assumptions: On the one hand, we have multikey FHE schemes without a setup, but with unstructured decryption circuit [LTV12] (and therefore not admitting a 1-round decryption procedure). On the other hand, we know of schemes that admit a 1-round decryption protocol, but they all require a trusted setup [CM15, MW16, BP16, PS16].

In this work, we ask the *reverse question*: Can existing 2-round MPC protocols help us constructing better multikey FHE schemes? The starting point of this work is the following observation. Let us assume for the moment that a secure 1-round MPC for all functions could exist. Then we could turn any multikey FHE in the plain model (e.g. that described in [LTV12]) into a multikey FHE with 1-round decryption, by simply evaluating the decryption circuit (no

[1] A description of both Ring LWE and DSPR assumptions can be found on pages 23 and 25, rescptively, of [LATV13] (version updated on 22 Oct 2014).

matter how complex it is) under the hood of the MPC. While this scenario is clearly unrealistic (1-round MPC is trivially impossible), it turns out the the intuition is still useful. Equipped with 2-round MPC, we can realize the above intuition as follows.

> *Encryption:* The encryption procedure consists of the encryption $\mathsf{Enc}(\mathsf{pk}, m)$ of a multikey FHE in the plain model (without 1-round decryption) and the first message of a 2-round MPC on input the secret key $\mathsf{sk}$.
>
> *Decryption:* Given an evaluated ciphertext $c$, each party computes locally the second message of the 2-round MPC for the function
>
> $$\Phi(\mathsf{sk}_1, \ldots, \mathsf{sk}_N) : \mathsf{Dec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_N), c)$$
>
> and broadcasts it. The plaintext can be publicly recovered by all parties by simply completing the execution of the MPC.

In the above outline we implicitly assumed that the 2-round MPC satisfies two properties: The scheme has (i) delayed function, i.e. the function to be computed can be specified in the second round and has (ii) public reconstruction, i.e. the output of the MPC can be publicly recovered given the transcript of the MPC. To instantiate this version of MPC we leverage recent works [GS18, BL18] that built 2-round MPC satisfying both properties[2] from the minimal assumption that 2-round oblivious transfer exists. It is important to observe that, although the communication complexity of the protocols in [ACGJ18, ACGJ19] grows polynomially with the size of the function evaluated, this does not affect the compactness of the resulting multikey FHE. This is because the size of the function $\Phi$ computed by the MPC is fixed, and in particular is independent of the size of the function that is homomorphically evaluated.

Unfortunately, the above construction does not quite achieve the security requirements for a multikey FHE. The reason is that all known 2-round MPCs in the plain model are *non-reusable*, i.e. the first message of the MPC can be used to securely compute a *single function*. This implies that the 1-round decryption procedure can be run exactly once per ciphertext, which is not what we would expect from a multikey FHE. Thus, the question of constructing multikey FHE in the plain model boils down to building reusable 2-round MPC, also in the plain model.

**Reusable MPC.** The recent work of Ananth et al. [AJJ20] on multiparty homomorphic encryption provides a construction of 2-round reusable MPC in the plain model, based on the hardness of LWE. Loosely speaking, their work shows a transformation from a *non-reusable* 2-round MPC with *succinct first-round messages* to a *reusable* 2-round MPC. In this context, succinct first-round messages means that the size of first-round messages of the MPC (for all parties) does not in any way depend on the size of the function being computed, and in particular is bounded by some polynomial $\mathsf{poly}(\lambda, N)$, where $\lambda$ is the security parameter and $N$ is the number of parties.

In the same work, [AJJ20] also provide a generic compiler to turn any 2-round MPC into a 2-round MPC with succinct first-round messages. However this transformation falls short of our requirements: in particular, it yields a *semi-honest* protocol even if we were to start with a *semi-malicious* MPC protocol.[3] Recall that semi-malicious security is a strengthening of the semi-honest definition and requires security to hold for well-formed messages but for

[2] Although the schemes were not presented with the delayed function property, they can be easily modified to satisfy this notion. This fact was already observed and used in [ACGJ18, ACGJ19].

[3] This is not just an artifact of the proof, but one can find an explicit attack for the scheme in [AJJ20] if the adversary is allowed to choose its random coins.

*any choice* of the random coins. Semi-maliciousness is, in fact, the de-facto security notion for multikey FHE (in the plain model) as defined in [MW16] and it is often leveraged in the usage of (multikey) FHE as a building block in more complex protocols (e.g. [MW16, BHP17, BGJ$^+$17], to mention a few) where semi-malicious security is useful.

The remainder of this overview (and the technical bulk of this work) is, therefore, dedicated to constructing a (non-reusable) 2-round semi-malicious MPC with succinct first-round messages. Once we have such a protocol, we can apply the reusability transformation of [AJJ20] (which preserves semi-malicious security) to obtain a reusable 2-round semi-malicious MPC protocol.

Towards this end, we observe that the protocol described above (when instantiated with a non-reusable MPC) *almost* achieves the succinct first-round messages property. The main challenge is that the size of the first message grows additionally with (i) the size of the inputs and (ii) the size of the output of the function being evaluated. We are going to overcome these challenges via generic transformations to our 2-round MPC.

**Removing Input Dependency.** It turns out that the issue of input dependency can be resolved with a very simple trick: Instead of running our MPC on its input $x$, we set the input to be the seed of a PRG $\mathsf{seed} \in \{0,1\}^\lambda$. Then, instead of evaluating $\Phi$ on $(x_1, \ldots, x_n)$, the MPC computes

$$\tilde{\Phi}_{(w_1,\ldots,w_N)}(\mathsf{seed}_1, \ldots, \mathsf{seed}_N) = \Phi(\mathsf{PRG}(\mathsf{seed}_1) \oplus w_1, \ldots, \mathsf{PRG}(\mathsf{seed}_N) \oplus w_N)$$

where the values $w_i = \mathsf{PRG}(\mathsf{seed}) \oplus x_1$ are hardwired in the description of the function and are communicated along with the first message of the MPC by each party. Technically, the size of each $w_i$ still depends on the input size so the first message is not properly succinct. However, the observation is that these $(w_1, \ldots, w_N)$ are *reusable* across multiple executions, even if the underlying MPC is not. Therefore, such an MPC satisfies the structural requirements on the first messages imposed by [AJJ20], as far as the dependency on the inputs is concerned.

**Removing Output Dependency.** Removing the dependency with respect to the output size requires much more work. To this end, we introduce the notion of *split multikey FHE*, an extension of the recently introduced split FHE [BDGM20] in the multiparty settings. Split multikey FHE is a three round primitive, where all parties (i) establish the public parameters of the system in the first round of interaction (this is known in the literature as distributed setup). Then each party can (ii) sample a key pair locally and encrypt their message under their public key. Analogously to (plain) multikey FHE, any function can be publicly computed over ciphertexts under independent keys, without the need to decrypt. The crucial property of split mutikey FHE is that, given an evaluated ciphertext and their secret key, each party can compute locally a *small decryption hint*, of size much smaller than the one of the evaluated ciphertext. Collecting all the decryption hints of all parties, allow anyone to (iii) decrypt the evaluated ciphertext and recover the output of the computation.

For the moment we are going to assume that such a split multikey FHE exists and we are going to demonstrate how to use it to remove the output dependency for a 2-round MPC. We defer the outline on the construction of split multikey FHE to a later part of this overview. Equipped with this tool, we can generically remove the output dependency from our 2-round MPC as follows.

> *First Message:* The first message consists of the first message of a 2-round MPC (with delayed function) where the input of each party is set to the seed of a PRG $\mathsf{seed}_i$ together

with its real input $x_i$. Additionally, the parties send the first message of the split multikey FHE to establish the public parameters of the scheme $\mathsf{pp}$.

*Second Message:* Each party sample a key pair $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ and computes an encryption of its input $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)$, where the random coins for these operations are taken deterministically from $\mathsf{PRG}(\mathsf{seed}_i)$. The second message of each party consists of $(\mathsf{pk}_i, c_i)$ along with the second message of the MPC for the function $\Phi$, defined as follows.

$\Phi(\mathsf{seed}_1, x_1, \ldots, \mathsf{seed}_N, x_N)$ : The function recomputes the keys $((\mathsf{sk}_1, \mathsf{pk}_1), \ldots, (\mathsf{sk}_N, \mathsf{pk}_N))$ and the ciphertexts $(c_1, \ldots, c_N)$ for all parties, then it evaluates the function $f$ homomorphically $\tilde{c} \leftarrow \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), f, (c_1, \ldots, c_N))$. Finally it outputs all of the decryption hints $(h_1, \ldots, h_N)$ for $\tilde{c}$.

*Output Reconstruction:* The homomorphically evaluated ciphertext $\tilde{c}$ can be publicly reconstructed, given the second messages. Then one recovers the decryption hints $(h_1, \ldots, h_N)$ from the output of the MPC to decrypt $\tilde{c}$.

First observe that, although the first message of the underlying MPC might depend polynomially on the size of its output, the function $\Phi$ only computes the decryption hints. Since $(h_1, \ldots, h_N)$ are of size independent of the output $|f(x_1, \ldots, x_N)|$, it follows that the communication complexity of the resulting protocol does not depend on the size of the output. Clearly, in order to achieve security, the decryption hints must not reveal anything beyond $f(x_1, \ldots, x_N)$. Thus what is left to be shown, is how to construct a secure split multikey FHE.

**Split Multikey FHE.** In the following we show how to construct split multikey FHE, assuming the hardness of the learning with errors (LWE) problem. Before delving into the details of our construction we shall clarify the relation with [BDGM20], and in particular with the notion of program obfuscation [GGH$^+$13]. It was shown in [BDGM20] that split FHE alone suffices to construct general purpose obfuscation, although their construction introduces some new assumptions about the circular dependency of the cryptography primitives used.

In order to obtain a construction from (plain) LWE we are going to relax the requirement for split (multikey) FHE. Specifically, we are going to allow the parties to additionally output a large *auxiliary information* $\alpha$ (of size possibly proportional to the output size). The condition that we impose, is that the computation of $\alpha$ does *not* depend on the evaluated ciphertext. One can think of $\alpha$ as a one-time re-randomization factor for the evaluated ciphertext. Note that even this *relaxed* notion of split multikey FHE suffices to remove the output dependency from a 2-round MPC: Since the computation of $\alpha$ does not depend on $\tilde{c}$, the parties can simply output it together with the second message of the MPC. Note that the size of the auxiliary information $\alpha$ prevents us from using this primitive to construct obfuscation, but it is still useful to remove the output dependency of the *first message* of the MPC. The reason is that we are not *compressing* the size of the output, but merely delaying it by one round.

**Augmenting [BHP17].** Our starting point is the multikey FHE scheme with distributed setup introduced in [BHP17]. Although we are not aware of any direct method to compute short decryption hints, this scheme has some useful properties, which we recall in the following. The scheme admits a 1-round setup, where all parties contribute to construct the public parameters and it admits multikey homomorphic evaluation of any polynomially computable function. The resulting ciphertext $\mathbf{C}$ defines a linear function $L_{\mathbf{C}}$ such that

$$L_{\mathbf{C}}(\mathbf{s}_1, \ldots, \mathbf{s}_N) = q/2 \cdot f(x_1, \ldots, x_N) + e \ (mod\ q)$$

where $q$ is the LWE modulus, $\mathbf{s}_i \in \{0,1\}^\eta$ (for some $\eta \in \mathsf{poly}(\lambda, N)$) are the secret keys of each party, and the variable $e$ is a bounded noise term. Note that, since $L_{\mathbf{C}}$ is a linear function, we can equivalently define the linear functions $L^{(1)}_{\mathbf{C}} \dots L^{(N)}_{\mathbf{C}}$ such that

$$\sum_{i=1}^{N} L^{(i)}_{\mathbf{C}}(\mathbf{s}_i) \approx q/2 \cdot f(x_1, \dots, x_N).$$

Intuitively, we are going to use the linear structure of the decryption algorithm to key-switch into a scheme for which we can define short decryption hints. Specifically, letting $\ell = |f(x_1, \dots, x_N)|$ be the size of the output of $f$, we are going to augment each public key of the scheme from [BHP17] with the following elements

$$(\mathbf{A}_i, \mathbf{B}_i = \mathbf{A}_i \cdot \mathbf{R}_i + \mathbf{E}_i + \mathbf{s}_i \otimes \mathbf{G})$$

where $\mathbf{A}_i \leftarrow\!\!\$\, \mathbb{Z}_q^{\ell \times n}$ is a uniform matrix, $\mathbf{R}_i$ corresponds to the LWE secret key, and $\mathbf{E}_i$ is the LWE noise, both of appropriate dimensions. Here $\mathbf{G}$ denotes the gadget matrix [MP12] and we define $\mathbf{G}^{-1}$ to be the corresponding bit-decomposition operator. For the sake of this outline, we omit the precise definitions and we are just going to use the fact that

$$\mathbf{s}_i \otimes \mathbf{G} \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \dots, L^{(i)}_{\mathbf{C}_\ell}\right) = \left(\mathbf{s}_i^T L^{(i)}_{\mathbf{C}_1}, \dots, \mathbf{s}_i^T L^{(i)}_{\mathbf{C}_\ell}\right) = \left(L^{(i)}_{\mathbf{C}_1}(\mathbf{s}_i), \dots, L^{(i)}_{\mathbf{C}_\ell}(\mathbf{s}_i)\right) \tag{1}$$

where $L^{(i)}_{\mathbf{C}_1} \dots L^{(i)}_{\mathbf{C}_\ell}$ are the vector representations of the linear functions defined by the evaluated ciphertexts $\mathbf{C}_1 \dots \mathbf{C}_\ell$. Before delving in the details of the computation of the decryption hints, observe that the elements $(\mathbf{A}_i, \mathbf{B}_i)$ essentially correspond to a (redundant) encryption of the secret key $\mathbf{s}_i$. Therefore the security of the augmented scheme follows from that of [BHP17], with an additional invocation of the LWE assumption.

**Decryption Hints and Auxiliary Information.** We are now ready to describe the computation of the decryption hints. First observe that, given $\ell$ evaluated ciphertexts $\mathbf{C}_1 \dots \mathbf{C}_\ell$, one can key-switch them into encryptions under $\mathbf{R}_i$ by computing

$$\begin{aligned}
&\sum_{i=1}^{N} \mathbf{B}_i \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \dots, L^{(i)}_{\mathbf{C}_\ell}\right) \\
&\approx \sum_{i=1}^{N} \mathbf{A}_i \cdot \mathbf{R}_i \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \dots, L^{(i)}_{\mathbf{C}_\ell}\right) + \mathbf{s}_i \otimes \mathbf{G} \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \dots, L^{(i)}_{\mathbf{C}_\ell}\right) \\
&\approx \sum_{i=1}^{N} \mathbf{A}_i \cdot \mathbf{R}_i \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \dots, L^{(i)}_{\mathbf{C}_\ell}\right) + \left(L^{(i)}_{\mathbf{C}_1}(\mathbf{s}_i), \dots, L^{(i)}_{\mathbf{C}_\ell}(\mathbf{s}_i)\right) \\
&\approx \sum_{i=1}^{N} \mathbf{A}_i \cdot \mathbf{R}_i \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \dots, L^{(i)}_{\mathbf{C}_\ell}\right) + f(x_1, \dots, x_N)
\end{aligned}$$

omitting small noise terms and applying Equation 1. Note that $\mathbf{A}_i$ is a publicly available matrix, so it is tempting to set the decryption hint $h_i$ of the $i$-th party to

$$h_i = \mathbf{R}_i \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \dots, L^{(i)}_{\mathbf{C}_\ell}\right) \in \mathbb{Z}_q^n.$$

While this would satisfy the compactness requirement ($|h_i|$ is independent of $\ell$), it turns out that this attempt is completely insecure since it reveals some non-trivial information about the secret key $\mathbf{R}_i$. To amend this, we define the decryption hint to be

$$h_i = \mathbf{R}_i \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \dots, L^{(i)}_{\mathbf{C}_\ell}\right) + \boldsymbol{\rho}_i$$

where $\boldsymbol{\rho}_i \leftarrow\!\!\$\, \mathbb{Z}_q^n$ is a uniformly sampled mask. At this point the hint is uniformly distributed over $\mathbb{Z}_q^n$ but we need to absorb the shift introduced by the mask $\boldsymbol{\rho}_i$. This is done by setting the auxiliary information $\alpha_i$ to

$$\alpha_i = \mathbf{A}_i \cdot \boldsymbol{\rho}_i + \mathbf{e}_i \in \mathbb{Z}_q^\ell$$

where the noise term $\mathbf{e}_i$ is introduced to be able to simulate the decryption hints and the auxiliary information in the security proof. Observe that, while the size of $\alpha_i$ is proportional to $\ell$, its computation is independent of the evaluated ciphertexts, as desired. A routine calculation shows that the hints $(h_1, \ldots, h_N)$ and the auxiliary information $(\alpha_1, \ldots, \alpha_N)$ suffice to recover the output of the function $f(x_1, \ldots, x_N)$ from the evaluated ciphertexts, for an appropriate set of parameters. This concludes the description of our split multikey FHE.

**Putting Things Together.** We conclude by summarizing the conceptual steps that are taken to construct our multikey FHE in the plain model.

**Step I:** Combine a non-reusable 2-round MPC (e.g. [GS18, BL18]) with a multikey FHE with unstructured decryption (e.g. [LTV12]) to obtain a non-reusable 2-round MPC where the size of the fist message depends on:

- The size of the inputs.
- The size of the output.
- The number of parties.
- The security parameter.

**Step II:** Apply a generic transformation to make the size of the first message independent of the size of the inputs.

**Step III:** Remove the dependency of the first message from the size of the output via split multikey FHE with distributed setup.

**Step IV:** Apply the compiler from [AJJ20] to obtain a reusable 2-round MPC with communication complexity proportional to the size of the function.

**Step V:** Combine (again!) the resulting reusable 2-round MPC with the multikey FHE of [LTV12] to obtain a multikey FHE in the plain model with 1-round decryption.

### 1.3 Perspectives and Open Problems

One of the main technical innovations of our work is the concept of split multikey FHE, which we believe to be an object of independent interest. For example, it naturally yields a 2-round MPC protocol with a input- and function-independent pre-processing with the following features: (i) The pre-processing is non-interactive (consists of a single round of broadcast among all parties) and its size depends only on the size of the output and on the security parameter. Furthermore (ii) the online communication depends on the size of the inputs, the number of parties, and the security parameter, and in particular is *independent* of the size of the output. Note that the pre-processing is not reusable, but this is the best that we can hope for without implying some form of obfuscation [HW15].

Our work leaves open some interesting directions for future research. The most compelling problem is to construct a multikey FHE with 1-round decryption assuming only the hardness of the (plain) learning with errors (LWE) problem. Another relevant direction is to improve the practical efficiency of our proposal and to obtain a more direct construction of multikey FHE from lattice assumptions.

# 2 Preliminaries

We denote by $\lambda \in \mathbb{N}$ the security parameter. We we say that a function $\mathsf{negl}(\cdot)$ is negligible if it vanishes faster than any polynomial. Given a set $S$, we denote by $s \leftarrow\!\!\$\, S$ the uniform sampling from $S$. We say that an algorithm is PPT if it can be implemented by a probabilistic Turing machine running in time $\mathsf{poly}(\lambda)$. Matrices are denoted by $\mathbf{M}$ and vectors are denoted by $\mathbf{v}$. We denote the infinity norm of a vector by $\|\mathbf{v}\|_\infty$. We recall the smudging lemma from [AIK11, AJL+12].

**Lemma 1** (Smudging)**.** *Let $B_1 = B_1(\lambda)$ and $B_2 = B_2(\lambda)$ be positive integers and let $e_1 \in \{1, \ldots, B_1\}$ be a fixed integer. Let $e_2 \leftarrow\!\!\$\, \{1, \ldots, B_2\}$ chosen uniformly at random. Then the distribution of $e_2$ is statistically indistinguishable from that of $e_2 + e_1$ as long as $B_1/B_2 = \mathsf{negl}(\lambda)$.*

## 2.1 Learning with Errors

We recall the (decisional) learning with errors (LWE) problem as introduced by Regev [Reg05].

**Definition 2.1** (Learning with Errors)**.** *The LWE problem is parametrized by a modulus $q$, positive integers $n, m$ and an error distribution $\chi$. The LWE problem is hard if the following distributions are computationally indistinguishable:*

$$(\mathbf{A}, \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}) \approx (\mathbf{A}, \mathbf{u})$$

*where $\mathbf{A}$ is chosen uniformly from $\mathbb{Z}_q^{n \times m}$, $\mathbf{s}$ is chosen uniformly from $\mathbb{Z}_q^n$, $\mathbf{u}$ is chosen uniformly from $\mathbb{Z}_q^m$ and $\mathbf{e}$ is chosen from $\chi^m$.*

As shown in [Reg05, PRS17], for *any* sufficiently large modulus $q$ the LWE problem where $\chi$ is a discrete Gaussian distribution with parameter $\sigma = \beta q \geq 2\sqrt{n}$ (i.e. the distribution over $\mathbb{Z}$ where the probability of $x$ is proportional to $e^{-\pi(|x|/\sigma)^2}$), is at least as hard as approximating (for a quantum algorithm) the shortest independent vector problem (SIVP) to within a factor of $\gamma = \tilde{O}(n/\beta)$ in *worst case* dimension $n$ lattices. We refer to $\beta = \sigma/q$ as the *modulus-to-noise* ratio, and by the above this quantity controls the hardness of the LWE instantiation. LWE with super-polynomial $\beta$ is conjectured to be hard for all PPT machines. We can truncate the discrete gaussian distribution $\chi$ to $\sigma \cdot \omega(\sqrt{\log(\lambda)})$ while only introducing a negligible gap between the distributions. Consequently, we omit the actual distribution $\chi$ but only use the fact that it can be bounded by some (small) value $B_\chi$.

## 2.2 Pseudorandom Generators

A pseudorandom generator (PRG) [ILL89] stretches random strings into strings that look uniformly random to a computationally bounded adversary.

**Definition 2.2** (Pseudorandom Generator)**.** *A function $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^\theta$ is pseudorandom if $\theta > \lambda$ and if the following distributions are computationally indistinguishable:*

$$r \approx \mathsf{PRG}(\mathsf{seed})$$

*where $r \leftarrow\!\!\$\, \{0,1\}^m$ and $\mathsf{seed} \leftarrow\!\!\$\, \{0,1\}^\lambda$.*

## 2.3 Multi-Party Computation Protocols

In the following we recall the definition of multi-party computation (MPC) protocols [Gol09]. In this work we consider MPC protocols where all communications happen through a broadcast channel and that consist only in two rounds of interaction among parties. Furthermore, we require that they satisfy the following properties:

(1) *Public Reconstruction:* At the end of the second round of the protocol, the output of the computation can be publicly reconstructed based on the transcript of the protocol.

(2) *Delayed Function:* The computation of the first message does not depend on the function $f$ but only on the inputs.

Note that we can assume that condition (1) holds without loss of generality, since any MPC protocol can be modified to achieve this property in a natural way. So the only non-trivial requirement is condition (2). All known 2-round (semi-honest) MPC based on the minimal assumption of 2-round (semi-honest) oblivious transfer [GS18, BL18] can be modified to achieve this property, as discussed in [ACGJ18, ACGJ19].

**Definition 2.3** (MPC Protocol)**.** *A 2-round protocol* $\Pi$ *for a function family* $\mathcal{F}$ *is a tuple of deterministic polynomial-time algorithms* $(\mathsf{First}, \mathsf{Second}, \mathsf{Output})$ *defined as follows.*

$\mathsf{First}_i\left(1^\lambda, x_i, s_i\right)$ : *On input the security parameter* $1^\lambda$*, an input* $x_i \in \{0,1\}^k$*, and a random tape* $s_i \in \{0,1\}^s$*, return the first round message* $y_i$ *for the party* $P_i$*.*

$\mathsf{Second}_i\left(1^\lambda, x_i, s_i, f, (y_1, \ldots, y_N)\right)$ : *On input the security parameter* $1^\lambda$*, an input* $x_i \in \{0,1\}^k$*, a random tape* $s_i \in \{0,1\}^s$*, the circuit representation of a function* $f$*, and the first round messages* $(y_1, \ldots, y_N)$*, return the second round message* $z_i$ *for the party* $P_i$*.*

$\mathsf{Output}\left(1^\lambda, (z_1, \ldots, z_N)\right)$ : *On input the security parameter* $1^\lambda$ *and the second round messages* $(z_1, \ldots, z_N)$*, return the output of the computation.*

The MPC protocol must satisfy correctness in the following sense.

**Definition 2.4** (Correctness)**.** *For all integers* $\lambda \in \mathbb{N}$*, all inputs* $(x_1, \ldots, x_N) \in \{0,1\}^{k \cdot N}$*, all functions* $f \in \mathcal{F}$*, and all random tapes* $(s_1, \ldots, s_N) \in \{0,1\}^{s \cdot N}$ *it holds that*

$$\mathsf{Output}\left(1^\lambda, (z_1, \ldots, z_N)\right) = f(x_1, \ldots, x_N)$$

*where* $z_i = \left\{\mathsf{Second}_i\left(1^\lambda, x_i, s_i, f, (y_1, \ldots, y_N)\right)\right\}_{i \in [N]}$ *and* $y_i = \left\{\mathsf{First}_i\left(1^\lambda, x_i, s_i\right)\right\}_{i \in [N]}$.

We now define security for MPC. In this work we are interested in the *semi-malicious* notion of security, where the adversary is required to behave honestly but can choose the random coins of the corrupted parties arbitrarily. We first define the non-reusable variant of this notion, where security is required to hold for a single execution of the MPC protocol.

**Definition 2.5** (Semi-Malicious Security)**.** *Let* $N$ *be a positive integer and let* $\Pi$ *be a* 2*-round MPC protocol for a function family* $\mathcal{F}$*. Then* $\Pi$ *is secure against semi-malicious adversaries if there exists a PPT algorithm* $\mathsf{Sim}$ *such that for all* $\lambda \in \mathbb{N}$*, all* $I \subset \{1, \ldots, N\}$*, all inputs* $(x_1, \ldots, x_N)$*, all random tapes* $(s_1, \ldots, s_N) \in \{0,1\}^s$*, all functions* $f \in \mathcal{F}$*, it holds that the following distributions are computationally indistinguishable:*

$$\begin{aligned}\Big(f, \{x_i, s_i\}_{i \in I}, \left\{y_i = \mathsf{First}_i(1^\lambda, x_i, s_i), \mathsf{Second}_i\left(1^\lambda, x_i, s_i, f, (y_1, \ldots, y_N)\right)\right\}_{i \in N}\Big) \\ \approx \mathsf{Sim}\left(1^\lambda, I, \{x_i, s_i\}_{i \in I}, f, f(x_1, \ldots, x_N)\right)\end{aligned}$$

We then define semi-malicious security in the reusable settings, where the first message of the MPC is fixed and security is required to hold for arbitrarily (but polynomially) many instances of the second message.

**Definition 2.6** (Reusable Semi-Malicious Security)**.** *Let $N$ be a positive integer and let $\Pi$ be a 2-round MPC protocol for a function family $\mathcal{F}$. Then $\Pi$ is reusably secure against semi-malicious adversaries if there exists a PPT algorithm $\mathsf{Sim}$ and a negligible function $\mathsf{negl}$ such that for all $\lambda \in \mathbb{N}$ and all PPT algorithms $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ it holds that*

$$\left|\Pr[\mathsf{Exp}_{\mathcal{A},\mathsf{Real}}(1^\lambda) = 1] - \Pr[\mathsf{Exp}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the two experiments are defined as follows.*

$\underline{\mathsf{Exp}_{\mathcal{A},\mathsf{Real}}(1^\lambda)}$*:*

- $(I \subset \{1,\ldots,N\}, \{x_i\}_{i\in N}, \{s_i\}_{i\in I}, \mathsf{st}) \leftarrow \mathcal{A}_0(1^\lambda)$
- *Sample* $\{s_i \leftarrow\!\!\$\, \{0,1\}^s\}_{i\notin I}$ *and compute* $\{y_i \leftarrow \mathsf{First}_i(1^\lambda, x_i, s_i)\}_{i\in N}$
- *Return* $\mathcal{A}_1^{\mathcal{O}}(\mathsf{st}, \{y_i\}_{i\in N})$

*where $\mathcal{O}$ takes as input a function $f \in \mathcal{F}$ and returns* $\left\{\mathsf{Second}_i\left(1^\lambda, x_i, s_i, f, (y_1,\ldots,y_N)\right)\right\}_{i\notin I}$.

$\underline{\mathsf{Exp}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda)}$*:*

- $(I \subset \{1,\ldots,N\}, \{x_i\}_{i\in N}, \{s_i\}_{i\in I}, \mathsf{st}) \leftarrow \mathcal{A}_0(1^\lambda)$
- *Compute* $(\mathsf{st}', \{y_i\}_{i\in N}) \leftarrow \mathsf{Sim}_0\left(1^\lambda, I, \{x_i, s_i\}_{i\in I}\right)$
- *Return* $\mathcal{A}_1^{\mathcal{O}} \leftarrow (\mathsf{st}, \{y_i\}_{i\in N})$

*where $\mathcal{O}$ takes as input a function $f \in \mathcal{F}$, computes* $(\mathsf{st}'', \{z_i\}_{i\notin I}) \leftarrow \mathsf{Sim}_1(\mathsf{st}', f, f(x_1,\ldots,x_N))$*, sets* $\mathsf{st}' = \mathsf{st}''$*, and returns* $\{z_i\}_{i\notin I}$.

**On Public Reconstruction.** We now argue that any $N$-party MPC protocol $\Pi$ (possibly without public reconstruction) can be transformed into an $(N-1)$-party MPC protocol $\Pi'$ with public reconstruction. By the correctness of $\Pi$, there exists at least one party $P_i$ that obtains the output at the end of the interaction. Then $\Pi'$ proceeds as follows: On input a function $f$, the parties jointly execute $\Pi$ for the function $g$, defined as

$$g(x_1,\ldots,x_n) = f(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$$

i.e. it ignores the $i$-th input. The input $x_i$ and the state $s_i$ of $P_i$ are sampled by an arbitrary party $P_{j\neq i}$ and sent along with the first message of the protocol. Since the state and the input of $P_i$ are publicly known, its messages can be locally simulated by all parties. At the end of the interaction, the output of the computation can be publicly recovered, since $x_i$ and $s_i$ are publicly known. Correctness and (semi-honest) security directly carry over from $\Pi$.

## 2.4 Multikey Fully-Homomorphic Encryption

A multikey FHE allows one to compute functions over ciphertexts encrypted under different and independently sampled keys. One can then decrypt the result of the computation by gathering together the corresponding secret keys and run a decryption algorithm. In this work we explicitly distinguish between two families of schemes, depending on structural properties of the decryption algorithm.

(1) 1-*Round Decryption:* The decryption algorithm consists of two subroutines (i) a local phase ($\mathsf{PDec}$) where each party computes a decryption share of the ciphertext based only on its secret key and (ii) a public phase ($\mathsf{Rec}$) where the plaintext can be publicly reconstructed from the decryption shares. This variant is the focus of our work.

(2) *Unstructured Decryption:* The decryption is a (possibly interactive) protocol that takes as input a ciphertext and all secret keys and returns the underlying plaintext. No special structural requirements are imposed.

In this work we are interested in constructing the former, which has a wider range of applications, such as a natural 2-round MPC protocol with low communication. However, the latter is going to be a useful building block in our transformation.

**Definition 2.7** (Multikey Homomorphic Encryption)**.** *A multikey FHE scheme $\Psi$ for $N$ parties and for the function family $\mathcal{F}$ consists of the following efficient algorithms.*

$\mathsf{KeyGen}(1^\lambda)$ : *On input the security parameter $1^\lambda$, the key generation algorithm returns a key pair $(\mathsf{sk}, \mathsf{pk})$.*

$\mathsf{Enc}(\mathsf{pk}, m)$ : *On input a public key $\mathsf{pk}$ and a message $m$, the encryption algorithm returns a ciphertext $c$.*

$\mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), f, (c_1, \ldots, c_N))$ : *On input a vector of public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_N)$, an $N$-argument function $f \in \mathcal{F}$, and a vector of ciphertexts $(c_1, \ldots, c_N)$, the evaluation algorithm returns an evaluated ciphertext $c$.*

$\mathsf{Dec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_N), c)$ : *On input a vector of secret keys $(\mathsf{sk}_1, \ldots, \mathsf{sk}_N)$ and a ciphertext $c$, the decryption protocol returns a message $m$. We say that a multikey FHE has a* 1-round decryption *if the decryption protocol consists of the algorithms $\mathsf{PDec}$ and $\mathsf{Rec}$ with the following syntax.*

- $\mathsf{PDec}(\mathsf{sk}, c, i)$ : *The partial decryption algorithm takes as input a secret key $\mathsf{sk}$, a ciphertext $c$, and an index $i$, and returns a decryption share $\rho_i$.*
- $\mathsf{Rec}(\rho_1, \ldots, \rho_N)$ : *The reconstruction algorithm takes as input a set of decryption shares $(\rho_1, \ldots, \rho_N)$ and returns a message $m$.*

We say that the scheme is *fully* homomorphic if it is homomorphic for P/poly. In the following we define correctness for multikey FHE with 1-round decryption, the more general notion can be obtained by modifying our definition in a natural way. Note that we only define correctness for a single application of the homomorphic evaluation procedure, although one can define the more general notion of multi-hop correctness [GHV10].

**Definition 2.8** (Correctness)**.** *A multi-key homomorphic encryption scheme $\Psi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PDec}, \mathsf{Rec})$ for $N$ parties is correct if for all $\lambda \in \mathbb{N}$, all $N$-argument functions $f \in \mathcal{F}$, all inputs $(m_1, \ldots, m_N)$, all $(\mathsf{sk}_i, \mathsf{pk}_i)$ in the support of $\mathsf{KeyGen}(1^\lambda)$, and all ciphertexts $(c_1, \ldots, c_N)$ in the support of $(\mathsf{Enc}(\mathsf{pk}_1, m_1), \ldots, \mathsf{Enc}(\mathsf{pk}_N, m_N))$, it holds that*

$$\mathsf{Rec}(\rho_1, \ldots, \rho_N) = f(m_1, \ldots, m_N).$$

*where $\rho_i \leftarrow \mathsf{PDec}(\mathsf{sk}_i, \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), f, (c_1, \ldots, c_N)), i)$.*

We say that a scheme is *compact* if the size of the evaluated ciphertexts does not depend on the size of the circuit representation of $f$ and only grows with the security parameter (and possibly the number of keys $N$). Furthermore, we require that the runtime of the decryption algorithm (and of its subroutines $\mathsf{PDec}$ and $\mathsf{Rec}$) is independent of the size of the circuit representation of $f$. We now recall the standard notion of semantic security for public-key encryption [GM82].

**Definition 2.9** (Semantic Security)**.** *A public-key encryption scheme* $\Psi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PDec}, \mathsf{Rec})$ *is semantically secure if for all* $\lambda \in \mathbb{N}$ *and for all pairs of messages* $(m_0, m_1)$*, the following distributions are computationally indistinguishable:*

$$\mathsf{Enc}(\mathsf{pk}, m_0) \approx \mathsf{Enc}(\mathsf{pk}, m_1)$$

*where* $(\mathsf{sk}, \mathsf{pk}) \leftarrow_{\$} \mathsf{KeyGen}(1^\lambda)$*.*

We define the notion of simulation security for multikey FHE with 1-round decryption. Intuitively, this notion says that the decryption share do not reveal anything beyond the plaintext that they reconstruct to. In this work we present a unified notion that combines semantic security and *computational* indistingushability of partial decryption shares. This is a weakening of the definition given in [MW16], where the simulated decryption shares were required to be *statistically* close to the honestly compute ones. To the best of our knowledge, this weaker notion is sufficient for all applications of multikey FHE. Note that by default we consider a *semi-malicious* adversary, that is allowed to choose the random coins of the corrupted parties arbitrarily.

**Definition 2.10** (Simulation Security)**.** *A multi-key homomorphic encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PDec}, \mathsf{Rec})$ *is simulation secure if there exists a PPT algorithm* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ *and a negligible function* $\mathsf{negl}$ *such that for all* $\lambda \in \mathbb{N}$*, all polynomials* $N = N(\lambda)$*, and all PPT algorithms* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *it holds that*

$$\left|\Pr[\mathsf{Exp}_{\mathcal{A},\mathsf{Real}}(1^\lambda) = 1] - \Pr[\mathsf{Exp}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

*where the two experiments are defined as follows.*

$\underline{\mathsf{Exp}_{\mathcal{A},\mathsf{Real}}(1^\lambda)}$*:*

- $(I \subset \{1, \ldots, N\}, \{m_i\}_{i \in N}, \{r_i, \tilde{r}_i\}_{i \in I}, \mathsf{st}) \leftarrow \mathcal{A}_0(1^\lambda)$
- *Sample* $\{(r_i, \tilde{r}_i) \leftarrow_{\$} \{0,1\}^*\}_{i \notin I}$
- *Compute* $\{(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda; r_i)\}_{i \in N}$ *and* $\{c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m_i, \tilde{r_i})\}_{i \in N}$
- *Return* $\mathcal{A}_1^{\mathcal{O}}(\mathsf{st}, \{\mathsf{pk}_i, c_i\}_{i \in N})$

*where* $\mathcal{O}$ *takes as input a function* $f$*, computes* $c \leftarrow \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), f, (c_1, \ldots, c_N))$*, and returns* $\{\mathsf{PDec}(\mathsf{sk}_i, c, i)\}_{i \notin I}$*.*

$\underline{\mathsf{Exp}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda)}$*:*

- $(I \subset \{1, \ldots, N\}, \{m_i\}_{i \in N}, \{r_i, \tilde{r}_i\}_{i \in I}, \mathsf{st}) \leftarrow \mathcal{A}_0(1^\lambda)$
- *Compute* $(\mathsf{st}', \{\mathsf{pk}_i, c_i\}_{i \in N}) \leftarrow \mathsf{Sim}_0\left(1^\lambda, I, \{m_i, r_i, \tilde{r}_i\}_{i \in I}\right)$
- *Return* $\mathcal{A}_1^{\mathcal{O}} \leftarrow (\mathsf{st}, \{\mathsf{pk}_i, c_i\}_{i \in N})$

*where* $\mathcal{O}$ *takes as input a function* $f$*, computes* $(\mathsf{st}'', \{\rho_i\}_{i \notin I}) \leftarrow \mathsf{Sim}_1(\mathsf{st}', f, f(m_1, \ldots, m_N))$*, sets* $\mathsf{st}' = \mathsf{st}''$*, and returns* $\{\rho_i\}_{i \notin I}$*.*

# 3 Split Multikey FHE

In the following we define and construct split multikey FHE with distributed setup, which is going to be instrumental to improve on the communication complexity of MPC protocols.

### 3.1 Distributed Setup

We first relax the definition of multikey FHE to allow for a one-time setup that sets the public parameters that are going to be taken as input by all algorithms. We require that the public parameters $\mathsf{pp}$ consists of the concatenation of strings $(\mathsf{pp}_1, \ldots, \mathsf{pp}_N)$ each independently sampled by each party involved. We refer to this notion as distributed setup, and we recall the definitions from [BHP17].

**Definition 3.1** (Multikey FHE with Distributed Setup)**.** *A multikey FHE scheme $\Psi$ has a distributed setup if there exists the following efficient algorithm.*

$\mathsf{Setup}(1^\lambda, i)$ : *On input the security parameter $1^\lambda$ and an index $i$, the setup algorithm outputs a string $\mathsf{pp}_i$.*

*We then define the public parameters to $\mathsf{pp} = (\mathsf{pp}_1, \ldots, \mathsf{pp}_N)$ and we assume that all other algorithms take $\mathsf{pp}$ as an additional input.*

We require the scheme to satisfy semantic security and simulatability as defined before, except that they are required to hold for *all possible choices* of $\{\mathsf{pp}_i\}_{i \in I}$, where $I$ is the set of corrupted parties. Note that we consider adaptive choices of the adversarial public parameters, i.e. the corrupted parties can sample their $\mathsf{pp}_i$ depending on the honestly chosen $\{\mathsf{pp}_j\}_{j \notin I}$.

In the following we recall the salient features of a multikey FHE scheme with distributed setup proposed in [BHP17].

**Theorem 3.2** ([BHP17])**.** *Assuming the hardness of the LWE problem with super-polynomial modulo-to-noise ratio, there exists a multikey FHE with distributed setup such that the following conditions are satisfied.*

*(1) The public parameters are jointly sampled by all parties and consist of the concatenation of $N$ matrices $\mathbf{P}_1 \| \ldots \| \mathbf{P}_N$, where each matrix $\mathbf{P}_i \leftarrow_\$ \mathbb{Z}_q^{\eta \times n}$ is locally sampled by each party. The parameter $\eta$ is set to $\eta = (Nn+1)\log(q) + 2\lambda = \mathsf{poly}(\lambda, N)$.*

*(2) For any depth parameter $d$, the public key are of size $|\mathsf{pk}| = \mathsf{poly}(\lambda, N) \cdot d$, whereas the secret keys are binary vectors $\mathbf{s} \in \{0,1\}^\eta$, regardless of the depth parameter. Then there exists an evaluation algorithm $\mathsf{Eval}$ that given as input:*

- *(a) $N$ public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_N)$ supporting $d$-depth computations.*
- *(b) An $N$-argument boolean function function $f : (\{0,1\}^*)^N \to \{0,1\}$ whose circuit representation has depth at most $d$.*
- *(c) Encryptions of each argument $m_i$ under the corresponding public key $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m_i)$.*

*The $\mathsf{Eval}$ algorithm returns a matrix $\mathbf{C}$ that defines a linear function $L_{\mathbf{C}}$ such that*

$$L_{\mathbf{C}}(\mathbf{s}_1, \ldots, \mathbf{s}_N) = q/2 \cdot f(m_1, \ldots, m_N) + e \ (mod\ q)$$

*where $|e| < \tilde{B} = B_\chi \cdot \mathsf{poly}(\lambda)$. Note that, since $L_{\mathbf{C}}$ is a linear function we can equivalently define the linear functions $L_{\mathbf{C}}^{(1)} \ldots L_{\mathbf{C}}^{(N)}$ such that*

$$\sum_{i=1}^{N} L_{\mathbf{C}}^{(i)}(\mathbf{s}_i) = q/2 \cdot f(m_1, \ldots, m_N) + e \ (mod\ q).$$

*(3) By further making a circular-security assumption, there exists a scheme that supports evaluation of circuits of any depth without increasing the length of public keys.*

### 3.2 Split Decryption

We define the notion of multikey FHE with split decryption [BDGM20]. Loosely speaking, split decryption requires that the decryption procedure consists of a (i) local phase, where each party computes a *small decryption hint*, and a (ii) public phase, where the plaintext is recovered from the evaluated ciphertexts together with the decryption hints. In this work we consider a specific notion of split decryption, where the hint is further decomposed in two components:

(1) A short component $h$, of size independent of that of the evaluated ciphertext.

(2) An auxiliary information $\alpha$ without any bound on the size, but whose computation does not depend on the evaluated ciphertext, nor the secret key.

This relaxation of the notion will allow us to obtain a construction from the standard LWE problem. The formal definition is given in the following.

**Definition 3.3** (Multikey FHE with Split Decryption)**.** *A multikey FHE scheme $\Psi$ has split decryption if the decryption protocol consist of the algorithms* $\mathsf{Aux}$, $\mathsf{Hint}$, *and* $\mathsf{HRec}$ *with the following syntax.*

$\mathsf{Aux}(\mathsf{pk}, i)$ : *On input a public key* $\mathsf{pk}$ *and an index $i$ the auxiliary algorithm, returns an auxiliary information $\alpha_i$ and a state $\mathsf{st}_i$.*

$\mathsf{Hint}(\mathsf{sk}, c, \mathsf{st}, i)$ *On input the secret key* $\mathsf{sk}$*, a ciphertext $c$, a state* $\mathsf{st}$*, and an index $i$, the hint algorithm returns a decryption hint $h_i$.*

$\mathsf{HRec}(c, (h_1, \ldots, h_N), (\alpha_1, \ldots, \alpha_N))$ : *The reconstruction algorithm takes as input a ciphertext $c$, a set of decryption hints $(h_1, \ldots, h_N)$, and a set of auxiliary information $(\alpha_1, \ldots, \alpha_N)$, and returns a message $m$.*

The definition of correctness is adapted to the modified syntax in a natural way.

**Definition 3.4** (Correctness)**.** *A multi-key homomorphic encryption scheme with split decryption $\Psi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Aux}, \mathsf{Hint}, \mathsf{HRec})$ for $N$ parties is correct if for all $\lambda \in \mathbb{N}$, all $N$-argument functions $f \in \mathcal{F}$, all inputs $(m_1, \ldots, m_N)$, all $(\mathsf{sk}_i, \mathsf{pk}_i)$ in the support of $\mathsf{KeyGen}(1^\lambda)$, and all ciphertexts $(c_1, \ldots, c_N)$ in the support of $(\mathsf{Enc}(\mathsf{pk}_1, m_1), \ldots, \mathsf{Enc}(\mathsf{pk}_N, m_N))$, it holds that*

$$\mathsf{HRec}(c, (h_1, \ldots, h_N), (\alpha_1, \ldots, \alpha_N)) = f(m_1, \ldots, m_N).$$

*where $c \leftarrow \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), f, (c_1, \ldots, c_N))$, $h_i \leftarrow \mathsf{Hint}(\mathsf{sk}_i, c, \mathsf{st}_i, i)$, and $(\alpha_i, \mathsf{st}_i) \leftarrow \mathsf{Aux}(\mathsf{pk}_i, i)$.*

We require that the hints are *short*, i.e. their size does not depend on the size of the output $|f(m_1, \ldots, m_N)|$ and in particular is bounded by some fixed polynomial $\mathsf{poly}(\lambda)$. Note that we do not impose any structural requirement on the auxiliary information $\alpha$. The definition of semantic security is unchanged.

In the following we define the notion of simulatability for decryption hints, which demands that the decryption hints together with the auxiliary information allow one to decrypt only the designated ciphertext and do not reveal anything beyond that.

**Definition 3.5** (Simulation Security)**.** *A multi-key homomorphic encryption scheme with split decryption $\Psi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Aux}, \mathsf{Hint}, \mathsf{HRec})$ for $N$ parties is simulation secure if there exists a PPT algorithm* $\mathsf{Sim}$ *such that for all $\lambda \in \mathbb{N}$, all $I \subset \{1, \ldots, N\}$, all messages $(m_1, \ldots, m_N)$,*

*all functions $f \in \mathcal{F}$, all $(\mathsf{sk}_i, \mathsf{pk}_i)$ in the support of $\mathsf{KeyGen}(1^\lambda)$, all $c_i$ in the support of $\mathsf{Enc}(\mathsf{pk}_i, m_i)$, it holds that the following distributions are statistically indistinguishable:*

$$\begin{aligned}&\{\mathsf{Hint}(\mathsf{sk}_i, \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), f, (c_1, \ldots, c_N)), \mathsf{st}_i, i), \alpha_i\}_{i \notin I} \\ \approx\ &\mathsf{Sim}\left(1^\lambda, I, \{\mathsf{sk}_i\}_{i \in I}, \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), f, (c_1, \ldots, c_N)), f(m_1, \ldots, m_N)\right)\end{aligned}$$

*where $(\alpha_i, \mathsf{st}_i) \leftarrow \mathsf{Aux}(\mathsf{pk}_i, i)$ and the randomness is taken over the coins of the simulator $\mathsf{Sim}$ and the algorithms $\mathsf{Aux}$ and $\mathsf{Hint}$, and all other messages are fixed.*

### 3.3 Construction

In the following we describe our construction for a split multikey FHE with distributed setup, assuming the hardness of the LWE problem. As a building block for our scheme, we use the multikey FHE with distributed setup $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ introduced in [BHP17]. By Theorem 3.2, the existence of such a scheme is implied by the hardness of the LWE problem (with super-polynomial modulo-to-noise ratio). We now define the gadget matrix $\mathbf{G}$, which is going to be useful for our purposes.

**Definition 3.6** (Gadget Matrix [MP12])**.** *Let $\mathbf{g} = (1, 2, 4, \ldots, 2^{\lfloor \log(q) \rfloor})$ and let $\mathbf{g}^\eta$ be the column concatenation of $\eta$-many copies of $\mathbf{g}$, i.e. $\mathbf{g}^\eta = \underbrace{(\mathbf{g}, \ldots, \mathbf{g})}_{\eta}$. We define the gadget matrix $\mathbf{G}$ to be*

$$\mathbf{G} = \begin{pmatrix} \mathbf{g}^\eta & 0 & \ldots & 0 \\ 0 & \mathbf{g}^\eta & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \mathbf{g}^\eta \end{pmatrix} \in \mathbb{Z}_q^{\ell \times \eta\ell\lceil \log(q) \rceil}.$$

*We also define $\mathbf{G}^{-1} : \mathbb{Z}_q^{\eta\ell} \to \mathbb{Z}_q^{\eta\ell\lceil \log(q) \rceil}$ to be the binary decomposition operator.*

For notational convenience we denote

$$\mathbf{x} \otimes \mathbf{G} = \begin{pmatrix} (x_1 \cdot \mathbf{g}, \ldots, x_\eta \cdot \mathbf{g}) & 0 & \ldots & 0 \\ 0 & (x_1 \cdot \mathbf{g}, \ldots, x_\eta \cdot \mathbf{g}) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & (x_1 \cdot \mathbf{g}, \ldots, x_\eta \cdot \mathbf{g}) \end{pmatrix}.$$

where $\mathbf{x} \in \{0,1\}^\eta$. Let $\bar{L}$ be the (column) vector representation of $\ell$ linear functions $(L_1, \ldots, L_\ell) \in \mathbb{Z}_q^{\eta\ell}$. We are going to use the fact that

$$\mathbf{x} \otimes \mathbf{G} \cdot \mathbf{G}^{-1}(\bar{L}) = \left(\mathbf{x}^T L_1, \ldots, \mathbf{x}^T L_\ell\right) = \left(L_1(\mathbf{x}), \ldots, L_\ell(\mathbf{x})\right). \tag{2}$$

We describe our construction below for functions with $\ell$-bits of outputs $f : (\{0,1\}^*)^N \to \{0,1\}^\ell$. Throughout the following description we assume that $q$ is an even number, although the construction can be easily adapted to arbitrary moduli.

**Setup:** On input the security parameter $1^\lambda$, the setup algorithm returns $\mathbf{P}_i \leftarrow \mathsf{Setup}(1^\lambda, i)$. We define the public parameters of the scheme to be $\mathsf{pp} = \mathbf{P}_1 \| \ldots \| \mathbf{P}_N$.

**Key Generation:** On input the public parameters $\mathsf{pp}$, the key generation algorithm samples a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ and parses $\mathsf{sk} = \mathbf{s} \in \{0,1\}^\eta$. Then it samples two uniform matrices $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{\ell \times n}$ and $\mathbf{R} \leftarrow_\$ \mathbb{Z}_q^{n \times \eta\ell\lceil \log(q) \rceil}$ and a matrix $\mathbf{E} \leftarrow_\$ \chi^{\ell \times \eta\ell\lceil \log(q) \rceil}$ from the error distribution. The secret key is set to $\mathbf{R}$ and the public key to:

$$(\mathsf{pk}, \mathbf{A}, \mathbf{B} = \mathbf{A} \cdot \mathbf{R} + \mathbf{E} + \mathbf{s} \otimes \mathbf{G}).$$

**Encryption:** On input a public key $(\mathsf{pk}, \mathbf{A}, \mathbf{B})$ and a message $m$, the encryption algorithm computes and returns $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$.

**Evaluation:** On input a set of public keys $((\mathsf{pk}_1, \mathbf{A}_1, \mathbf{B}_1), \ldots, (\mathsf{pk}_N, \mathbf{A}_N, \mathbf{B}_N))$, an $\ell$-bit output function $f$ and a set of ciphertexts $(c_1, \ldots, c_N)$, the evaluation algorithm defines $f_i$ as the function that returns the $i$-th output bit of $f$ and computes

$$\mathbf{C}_i \leftarrow \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), f_i, (c_1, \ldots, c_N))$$

for all $i \in \{1, \ldots, \ell\}$. The algorithm returns $(\mathbf{C}_1, \ldots, \mathbf{C}_\ell)$.

**Auxiliary:** On input a public key $(\mathsf{pk}, \mathbf{A}, \mathbf{B})$ and an index $i$, the auxiliary algorithm samples a uniform $\boldsymbol{\rho} \leftarrow_\$ \mathbb{Z}_q^n$ and sets the state $\mathsf{st} = \boldsymbol{\rho}$. Then it computes the auxiliary information as

$$\alpha = \mathbf{A}\boldsymbol{\rho} + \tilde{\mathbf{e}}$$

where $\tilde{\mathbf{e}} \leftarrow_\$ \{-B, \ldots, B\}^\ell$.

**Hint:** On input a secret key $\mathbf{R}$, an $\ell$-bits evaluated ciphertext $(\mathbf{C}_1, \ldots, \mathbf{C}_\ell)$, a state $\boldsymbol{\rho}$, and an index $i$, the hint algorithm computes

$$h = \mathbf{R} \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \ldots, L^{(i)}_{\mathbf{C}_\ell}\right) + \boldsymbol{\rho}$$

where $L^{(i)}_{\mathbf{C}_j}$ is the vector representation of the $i$-th linear function induced by the $j$-th evaluated ciphertext.

**Reconstruction:** On input an $\ell$-bits evaluated ciphertext $(\mathbf{C}_1, \ldots, \mathbf{C}_\ell)$, a set of hints $(h_1, \ldots, h_N)$, and a set of auxiliary information $(\alpha_1, \ldots, \alpha_N)$, the reconstruction algorithm sets $L^{(i)}_{\mathbf{C}_j}$ to be the vector representation of the $i$-th linear function induced by the $j$-th evaluated ciphertext. Then it computes

$$\tilde{\mathbf{m}} = \sum_{i=1}^{N} \mathbf{B}_i \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \ldots, L^{(i)}_{\mathbf{C}_\ell}\right) - \mathbf{A}_i \cdot h_i + \alpha_i$$

and rounds each component to the nearest multiple of $q/2$ to obtain $\lfloor \tilde{\mathbf{m}} \rceil$. The algorithm returns $2/q \cdot \lfloor \tilde{\mathbf{m}} \rceil$.

### 3.4 Analysis

We first argue that our scheme as described above satisfies correctness.

**Theorem 3.7** (Correctness)**.** *Let* $q/4 > \tilde{B} + N \cdot (B_\chi \cdot \ell \cdot \eta \cdot \lceil \log(q) \rceil + B)$*. Then the scheme as*

*described above is correct.*

*Proof.* Consider the vector $\tilde{\mathbf{m}}$ as computed in the reconstruction algorithm. Then we substitute

$$\begin{aligned}\tilde{\mathbf{m}} &= \sum_{i=1}^{N} \mathbf{B}_i \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \ldots, L^{(i)}_{\mathbf{C}_\ell}\right) - \mathbf{A}_i \cdot h_i + \alpha_i \\ &= \sum_{i=1}^{N} (\mathbf{A}_i \cdot \mathbf{R}_i + \mathbf{E}_i + \mathbf{s}_i \otimes \mathbf{G}) \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \ldots, L^{(i)}_{\mathbf{C}_\ell}\right) - \mathbf{A}_i \cdot h_i + \alpha_i \\ &= \sum_{i=1}^{N} (\mathbf{A}_i \cdot \mathbf{R}_i + \mathbf{E}_i) \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \ldots, L^{(i)}_{\mathbf{C}_\ell}\right) + \left(L^{(i)}_{\mathbf{C}_1}(\mathbf{s}_i), \ldots, L^{(i)}_{\mathbf{C}_\ell}(\mathbf{s}_i)\right) - \mathbf{A}_i \cdot h_i + \alpha_i\end{aligned}$$

by the Equation 2. By the definition of $h_i$ we have that

$$\mathbf{A}_i \cdot h_i = \mathbf{A}_i \cdot \left(\mathbf{R}_i \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \ldots, L^{(i)}_{\mathbf{C}_\ell}\right) + \boldsymbol{\rho}_i\right)$$

and therefore

$$\begin{aligned}\tilde{\mathbf{m}} &= \sum_{i=1}^{N} \mathbf{E}_i \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \ldots, L^{(i)}_{\mathbf{C}_\ell}\right) + \left(L^{(i)}_{\mathbf{C}_1}(\mathbf{s}_i), \ldots, L^{(i)}_{\mathbf{C}_\ell}(\mathbf{s}_i)\right) - \mathbf{A}_i \cdot \boldsymbol{\rho}_i + \alpha_i \\ &= \sum_{i=1}^{N} \underbrace{\mathbf{E}_i \cdot \mathbf{G}^{-1}\left(L^{(i)}_{\mathbf{C}_1}, \ldots, L^{(i)}_{\mathbf{C}_\ell}\right)}_{\mathbf{e}'_i} + \left(L^{(i)}_{\mathbf{C}_1}(\mathbf{s}_i), \ldots, L^{(i)}_{\mathbf{C}_\ell}(\mathbf{s}_i)\right) + \tilde{\mathbf{e}}_i\end{aligned}$$

since $\alpha_i = \mathbf{A}_i\boldsymbol{\rho}_i + \tilde{\mathbf{e}}_i$. By the correctness of the multikey FHE with distributed setup (Theorem 3.2) we obtain

$$\begin{aligned}\tilde{\mathbf{m}} &= \sum_{i=1}^{N} \left(L^{(i)}_{\mathbf{C}_1}(\mathbf{s}_i), \ldots, L^{(i)}_{\mathbf{C}_\ell}(\mathbf{s}_i)\right) + \mathbf{e}'_i + \tilde{\mathbf{e}}_i \\ &= q/2 \cdot (f_1(m_1, \ldots, m_N), \ldots, f_\ell(m_1, \ldots, m_N)) + \mathbf{e} + \sum_{i=1}^{N} \mathbf{e}'_i + \tilde{\mathbf{e}}_i \\ &= q/2 \cdot (f_1(m_1, \ldots, m_N), \ldots, f_\ell(m_1, \ldots, m_N)) + \mathbf{e} + \mathbf{e}' + \tilde{\mathbf{e}}\end{aligned}$$

where $\|\mathbf{e}\|_\infty \leq \tilde{B}$. On the other hand, by definition of $\mathbf{E}$ we have that $\|\mathbf{e}'\|_\infty \leq N \cdot B_\chi \cdot \ell \cdot \eta \cdot \lceil \log(q) \rceil$, and by definition of $\alpha$ we have that $\|\tilde{\mathbf{e}}\|_\infty \leq N \cdot B$. Thus, as long as $q/4 > \tilde{B} + N \cdot (B_\chi \cdot \ell \cdot \eta \cdot \lceil \log(q) \rceil + B)$, correctness always holds. □

We now show that our construction satisfies the structural requirements for a split multikey FHE (with distributed setup). This is easily proven by the fact that the hint of each party $h$ is an $n$-dimensional vector in $\mathbb{Z}_q^n$ and in particular is independent of $\ell$, the size of the output of $f$.

Next we show that our scheme satisfies the notion of simulatability for the decryption hints. Note that we show only simulatability against a maximally corrupted subset of $N-1$ parties, similarly as in [MW16].

**Theorem 3.8** (Simulation Security)**.** *Let* $\left(\tilde{B} + B_\chi \cdot \ell \cdot \eta \cdot \lceil \log(q) \rceil\right) \cdot B^{-1} = 2^{-\lambda}$. *Then the scheme as described above is simulation secure.*

*Proof.* On input the secret keys $\{\mathbf{s}_i\}_{i\in I}$ of the corrupted parties, the evaluated ciphertext $(\mathbf{C}_1,\ldots,\mathbf{C}_\ell)$, and the function output $(\mu_1,\ldots,\mu_\ell) = f(m_1,\ldots,m_\ell)$, the simulator samples a uniform $h_j \leftarrow_\$ \mathbb{Z}_q^n$, where $j \notin I$ is the index of the honest party. Then it computes the auxiliary information as

$$\tilde{\alpha}_j = q/2\cdot(\mu_1,\ldots,\mu_\ell) - \sum_{i\in I}\left(L^{(i)}_{\mathbf{C_1}}(\mathbf{s}_i),\ldots,L^{(i)}_{\mathbf{C}_\ell}(\mathbf{s}_i)\right) - \mathbf{B}_j\cdot\mathbf{G}^{-1}\left(L^{(j)}_{\mathbf{C}_1},\ldots,L^{(j)}_{\mathbf{C}_\ell}\right) + \mathbf{A}_j\cdot h_j + \tilde{\mathbf{e}}$$

where $\tilde{\mathbf{e}} \leftarrow_\$ \{-B,\ldots,B\}^\ell$. Recall that, by the correctness of the multikey FHE with distributed setup (Theorem 3.2), it holds that

$$\sum_{i=1}^{N}\left(L^{(i)}_{\mathbf{C_1}}(\mathbf{s}_i),\ldots,L^{(i)}_{\mathbf{C}_\ell}(\mathbf{s}_i)\right) = q/2\cdot(\mu_1,\ldots,\mu_\ell) + \mathbf{e}.$$

Furthermore, the proof of Theorem 3.7 shows that

$$\mathbf{A}_j\cdot h_j = \alpha_j - \tilde{\mathbf{e}} + \mathbf{B}_j\cdot\mathbf{G}^{-1}\left(L^{(j)}_{\mathbf{C}_1},\ldots,L^{(j)}_{\mathbf{C}_\ell}\right) - \mathbf{e}'_j - \left(L^{(j)}_{\mathbf{C_1}}(\mathbf{s}_j),\ldots,L^{(j)}_{\mathbf{C}_\ell}(\mathbf{s}_j)\right)$$

where $\alpha_j$ is the honestly computed auxiliary information. This means that the difference between the simulated $\tilde{\alpha}_j$ and the honest $\alpha_j$ is bounded by $\|\mathbf{e}+\mathbf{e}'_j\|_\infty$. Thus as long as the following distributions are statistically close

$$\tilde{\mathbf{e}} \approx \tilde{\mathbf{e}} + \mathbf{e} + \mathbf{e}'_j$$

then so is the output of the simulator. Since $\|\mathbf{e}+\mathbf{e}'_j\|_\infty \le \tilde{B} + B_\chi\cdot\ell\cdot\eta\cdot\lceil\log(q)\rceil$ and $\left(\tilde{B} + B_\chi\cdot\ell\cdot\eta\cdot\lceil\log(q)\rceil\right)\cdot B^{-1} = 2^{-\lambda}$, then the indistinguishability follows by Lemma 1. □

Finally we show that the scheme satisfies semantic security.

**Theorem 3.9** (Semantic Security)**.** *If the LWE with super-polynomial modulo-to-noise ratio is hard, then the scheme as described above is semantically secure.*

*Proof.* By a standard hybrid argument, we have that the matrix $\mathbf{B}$ is computationally indistinguishable from a uniform $\mathbf{U} \leftarrow_\$ \mathbb{Z}_q^{\ell\times\eta\ell\lceil\log(q)\rceil}$. Then semantic security follows from the security of the underlying multikey FHE with distributed setup (Theorem 3.2). □

### 3.5 Parameters

The above analysis fixes a set of constraints on the parameters. In the following we show assignments that satisfy these equations and lead to an instance of LWE which is conjectured to be hard. By correctness we require that

$$q/4 > \tilde{B} + N\cdot(B_\chi\cdot\ell\cdot\eta\cdot\lceil\log(q)\rceil + B)$$

whereas simulation security demands that

$$\frac{\tilde{B} + B_\chi\cdot\ell\cdot\eta\cdot\lceil\log(q)\rceil}{B} = 2^{-\lambda}.$$

Recall that $\tilde{B} = B_\chi\cdot\mathsf{poly}(\lambda)$. Fix some bound for the initial noise $B_\chi \in \mathsf{poly}(\lambda)$, then setting $B = 2^{O(\lambda\log(\lambda))}\cdot B_\chi$ and $q = 2^{\omega(\lambda\log(\lambda))}\cdot B_\chi$ we can ensure that both of the conditions are simultaneously satisfied. The parameter $n$ is free and we can take it to be large enough so that the corresponding LWE instance is conjectured to be hard.

# 4 Bootstrapping Multikey FHE

In the following we show a construction of a multikey FHE with 1 round decryption assuming the existence of the following tools:

(1) Two-round MPC $\Pi = (\mathsf{First}_i, \mathsf{Second}_i, \mathsf{Output})$ with delayed function and public reconstruction and

(2) Multikey FHE $\Psi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ with unstructured decryption.

We present our scheme below and we defer the discussion about plausible instantiations for the underlying building blocks to Section 6.

**Key Generation:** On input the security parameter $1^\lambda$, the key generation algorithm samples a random string $s_i \leftarrow\!\!\$\, \{0,1\}^s$ and a key pair $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$. Then it computes $y_i \leftarrow \mathsf{First}_i(1^\lambda, \mathsf{sk}_1, s_i)$. The algorithm sets the secret key to $\mathsf{sk} = (\mathsf{sk}_i, s_i)$ and the public key to $\mathsf{pk} = (\mathsf{pk}_i, y_i)$.

**Encryption:** On input a public key $\mathsf{pk} = (\mathsf{pk}_i, y_i)$ and a message $m$, the encryption algorithm computes and returns $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m)$.

**Evaluation:** On input a set of public keys $((\mathsf{pk}_1, y_1), \ldots, (\mathsf{pk}_N, y_N))$, a function $f$ and a set of ciphertexts $(c_1, \ldots, c_N)$, the evaluation algorithm computes and returns

$$c \leftarrow \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), f, (c_1, \ldots, c_N))$$

along with $(y_1, \ldots, y_N)$.

**Partial Decryption:** On input a secret key $\mathsf{sk} = (\mathsf{sk}_i, s_i)$, an evaluated ciphertext $c$, and an index $i$, the decryption algorithm computes and returns

$$z_i \leftarrow \mathsf{Second}_i(1^\lambda, \mathsf{sk}_i, s_i, g, (y_1, \ldots, y_N))$$

where $g$ is defined as

$$g(\mathsf{sk}_1, \ldots, \mathsf{sk}_N) = \mathsf{Dec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_N), c).$$

**Reconstruction:** On input the decryption shares $(z_1, \ldots, z_N)$, the reconstruction algorithm returns $\mathsf{Output}(1^\lambda, (z_1, \ldots, z_N))$.

We argue that the scheme satisfies correctness.

**Theorem 4.1** (Correctness)**.** *Let $\Pi$ be a correct 2-round MPC with public reconstruction and delayed function and let $\Psi$ be a correct multikey FHE with unstructured decryption. Then our scheme as described above is correct.*

*Proof.* Consider the shares $(z_1, \ldots, z_N)$ taken as input by the reconstruction subroutine. Substituting we have that

$$z_i = \mathsf{Second}_i(1^\lambda, \mathsf{sk}_i, s_i, g, (y_1, \ldots, y_N))$$

where

$$y_i = \mathsf{First}_i(1^\lambda, \mathsf{sk}_i, s_i).$$

By the correctness of the MPC we have that

$$\mathsf{Output}(1^\lambda, (z_1, \ldots, z_N)) = \mathsf{Dec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_N), c).$$

Invoking the correcntess of the multikey FHE we obtain

$$\begin{aligned}
&\mathsf{Dec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_N), c) \\
&= \mathsf{Dec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_N), \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), f, (c_1, \ldots, c_N))) \\
&= \mathsf{Dec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_N), \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), f, (\mathsf{Enc}(\mathsf{pk}_1, m_1), \ldots, \mathsf{Enc}(\mathsf{pk}_N, m_N)))) \\
&= f(m_1, \ldots, m_N).
\end{aligned}$$

as desired. □

We now argue that the scheme is compact. First observe that the evaluated ciphertexts consist of two components: (i) an evaluated ciphertext $c$ and (ii) the set of first messages $(y_1, \ldots, y_N)$. The former is clearly independent of the circuit size by the compactness of the underlying multikey FHE scheme, i.e. $|c| = \mathsf{poly}(\lambda, N, |f(m_1, \ldots, m_N)|)$. On the other hand the vector $(y_1, \ldots, y_N)$ grows with the size of the inputs an with the circuit representation of the function that is computed. This means that the size of the vector $(y_1, \ldots, y_N)$ depends polynomially on $N$, the size of the inputs $(\mathsf{sk}_1, \ldots, \mathsf{sk}_N)$ and the circuit representation of the function $g$. That is we have that $|(y_1, \ldots, y_N)| = \mathsf{poly}(\lambda, N, |f(m_1, \ldots, m_N)|)$. So the overall size of the evaluated ciphertexts is $\mathsf{poly}(\lambda, N, |f(m_1, \ldots, m_N)|)$ and in particular is independent of the size of $f$. Thus compactness follows. Next we show that the scheme is simulation secure. Depending on the instantiation of $\Pi$, we obtain two flavours of simulation security, i.e. reusable secure or not.

**Theorem 4.2** (Semantic Security)**.** *Let $\Pi$ be a semi-malicious (reusable, resp.) secure 2-round MPC with public reconstruction and delayed function and let $\Psi$ be a semantically secure multikey FHE with unstructured decryption. Then our scheme as described above is (reusable, resp.) simulation secure.*

*Proof.* Consider the following sequence of hybrid distributions.

*Hybrid $\mathcal{H}_0$:* This is the distribution induced by the real experiment.

*Hybrid $\mathcal{H}_1$:* This distribution is identical to the previous one, except that messages $\{y_i, z_i\}_{i \in I}$ of the honest parties are computed using the simulator of the MPC protocol. By the semi-malicious security of $\Pi$, the resulting distribution is computationally indistinguishable from that of the previous hybrid.

*Hybrid $\mathcal{H}_2$:* This distribution is identical to the previous one, except that the ciphertexts of the honest parties are computed as encryptions of 0, padded to the appropriate length. I.e. it holds that

$$\{c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, 0)\}_{i \notin I}$$

where $\mathsf{pk}$ is sampled with the procedure specified in $\mathcal{H}_1$. Indistinguishability follows from an invocation of semantic security of $\Psi$. The description of our simulator coincides with the distribution induced by this hybrid. Observe that if $\Pi$ is reusable secure, then so is the resulting multikey FHE. This completes our proof. □

# 5 Semi-Malicious Reusable MPC

In this section, we show how to construct a two-round MPC satisfying reusable semi-malicious security (Definition 2.6) in the following steps.

Our starting step is a delayed-function two-round semi-malicious non-reusable MPC with first round complexity independent of the complexity of the function being securely computed. We obtain this by applying the bootstrapping step described in Section 4 on any delayed-function two-round semi-malicious MPC with public reconstruction property.

We formalize this in the following theorem.

**Theorem 5.1.** *Assuming any multikey FHE assuming unstructured decryption (Section 2.4), there exists a delayed-function semi-malicious secure multiparty computation protocol for a $N$-party functionality satisfying the property that the computation of the first round messages of all the parties is a polynomial in $(\lambda, N, \ell_{\ell_{\mathsf{in}}}, \ell_{\ell_{\mathsf{out}}})$, where $\ell_{\mathsf{in}}$ is the input length of each party and $\ell_{\mathsf{out}}$ is the output length of the functionality. In particular, the complexity of computing the first round messages is independent of the complexity of the function being securely computed.*

Equipped with such a (non-reusable) semi-malicious MPC, we then show how to construct a semi-malicious MPC in the following steps.

**Step I:** First, we show how to transform the MPC satisfying the property stated in Theorem 5.1 into a two-round semi-malicious MPC with the following property:

- The first round messages are computed using two algorithms $\mathsf{Reuse.First}$ and $\mathsf{NonReuse.First}$ such that the time to compute $\mathsf{Reuse.First}$ is a polynomial in $(\lambda, N, \ell_{\mathsf{in}}, \ell_{\mathsf{out}})$ and the time to compute $\mathsf{NonReuse.First}$ is a polynomial in $(\lambda, N, \ell_{\mathsf{out}})$ (and in particular, independent of the input length). That is, the first round messages computed by the $i^{th}$ party is the concatenation of the outputs of $\mathsf{Reuse.First}_i$ and $\mathsf{NonReuse.First}_i$.

  Moreover, $\mathsf{Reuse.First}$ has the property that it is reusable for multiple executions of secure MPC (we will state this more formally in the theorem statements).

**Step II:** Next, we show how to further transform the semi-malicious non-reusable MPC obtained in the first step to obtain a semi-malicious non-reusable MPC satisfying the following property:

- The first round messages are computed using two algorithms $\mathsf{Reuse.First}$ and $\mathsf{NonReuse.First}$ such that the time to compute $\mathsf{Reuse.First}$ is a polynomial in $(\lambda, N, \ell_{\mathsf{in}})$ and the time to compute $\mathsf{NonReuse.First}$ is a polynomial in $(\lambda, N)$. In particular, the complexity of the first round messages is completely independent of the output length of the functionality being securely computed. As before, the first round messages computed by the $i^{th}$ party is the concatenation of the outputs of $\mathsf{Reuse.First}_i$ and $\mathsf{NonReuse.First}_i$.

**Step III:** Finally, we apply a modified version of reusability transformation of [AJJ20] on the semi-malicious non-reusable MPC obtained from the second step to obtain a two-round MPC satisfying reusable semi-malicious security (Definition 2.6).

## 5.1 Removing Input Dependency

In the following we show how to generically compile any MPC with delayed function and public reconstruction where the first message possibly depends on the input size, to an MPC with the same characteristic but where the first message is composed by two parts: (i) a non-reusable component that is independent of the size of the inputs and a (ii) reusable component that depends on the size of the input. While this transformation does not completely remove the input dependence, it suffices to build an MPC that can be used as a building block for Theorem 5.5. Our scheme builds on the following tools:

(1) Two-round MPC $\Pi = (\mathsf{First}_i, \mathsf{Second}_i, \mathsf{Output})$ with delayed function and public reconstruction satisfying the following property: the complexity of computing the first round messages is a polynomial in $\lambda$, input lengths of all parties, output length of the function and the number of parties.

(2) Cryptographic pseudorandom generator $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{\mathsf{poly}(\lambda)}$.

The scheme is described below.

**First Message:** On input the security parameter $1^\lambda$, the input $x$, and the random coins $s$, the first message algorithm samples (deterministically from $s$) a uniform $\mathsf{seed}$ and computes $w \leftarrow \mathsf{PRG}(\mathsf{seed}) \oplus x$. Then it returns

$$(w, \mathsf{First}_i(1^\lambda, \mathsf{seed}, s)).$$

**Second Message:** On input the security parameter $1^\lambda$, an input $x$, the random coins $s$, a function $f$, and the first messages $((w_1, y_1), \ldots, (w_N, y_N))$ the second message algorithm defines the function $g$ as

$$g(\mathsf{seed}_1, \ldots, \mathsf{seed}_N) = f(\mathsf{PRG}(\mathsf{seed}_1) \oplus w_1, \ldots, \mathsf{PRG}(\mathsf{seed}_N) \oplus w_N).$$

Then it recomputes $\mathsf{seed}$ from $s$ and returns $\mathsf{Second}_i(1^\lambda, \mathsf{seed}, s, g, (y_1, \ldots, y_N))$.

**Reconstruction:** On input the security parameter $1^\lambda$ and the second round messages $(z_1, \ldots, z_N)$, the output reconstruction algorithm returns $\mathsf{Output}(1^\lambda, (z_1, \ldots, z_N))$.

Note that the component $|w_i|$ in the first message does depend on the size of the input $|x_i|$ however it is reusable in the sense that it can be fixed for multiple (possibly independent) instances of the MPC protocol. Clearly the component $y_i = \mathsf{First}_i(1^\lambda, \mathsf{seed}_i, s_i)$ cannot depend on the size of the input. The following theorem shows that the transformation yeilds a secure MPC protocol.

**Theorem 5.2** (Input Independent MPC)**.** *Let $\Pi$ be a two-round MPC protocol satisfying the properties stated in Theorem 5.1, let $\mathsf{PRG}$ be a cryptographic pseudorandom generator, our scheme as described above is a two-round semi-malicious MPC satisfying the following properties:*

- *The first round messages are computed using two algorithms $\mathsf{Reuse.First}$ and $\mathsf{NonReuse.First}$ such that the time to compute $\mathsf{Reuse.First}$ is a polynomial in $(\lambda, \ell_{\mathsf{in}}, \ell_{\mathsf{out}})$ and the time to compute $\mathsf{NonReuse.First}$ is a polynomial in $(\lambda, \ell_{\mathsf{out}})$ (and in particular, independent of the input length). That is, the first round messages computed by the $i^{th}$ party is the concatenation of the outputs of $\mathsf{Reuse.First}_i$ and $\mathsf{NonReuse.First}_i$.*
- *Moreover, the algorithm $\mathsf{Reuse.First}$ is reusable across multiple executions of secure MPC: that is, in the offline phase, every party on input $x_i$ can compute $\mathsf{Reuse.First}$ and broadcast its output. In the online phase, executed polynomially many times, all the parties securely compute an $N$-party functionality on the same inputs $(x_1, \ldots, x_N)$ with the only difference being that the first round messages are computed using only $\mathsf{NonReuse.First}$; in particular the output of $\mathsf{Reuse.First}$ is reused for every execution of the online phase.*

*Proof.* Correctness follows trivially from the protocol inspection. To argue security we consider the following sequence of hybrid distributions.

*Hybrid $\mathcal{H}_0$:* This is the distribution where the messages of the MPC are honestly computed.

*Hybrid $\mathcal{H}_1$:* This distribution is identical to the previous one, except that the messages of $\Pi$ are computed using the corresponding simulator. By the semi-malicious security of $\Pi$, the resulting distribution is computationally indistinguishable from that of the previous hybrid.

*Hybrid $\mathcal{H}_2$:* This distribution is identical to the previous one, except that $w$ is a uniformly sampled string. This change is indistinguishable by the pseudorandomness if $\mathsf{PRG}$. This hybrid corresponds to the description of our simulator and completes the proof. □

### 5.2 Removing Output Dependency

In the following we show a generic transformation to compile any MPC whose first message depends on the output size into an MPC whose first message does not depend on the output size, while preserving the security of the scheme. Our construction assumes the existence of the following tools:

(1) Two-round MPC $\Pi = (\mathsf{First}_i, \mathsf{Second}_i, \mathsf{Output})$ satisfying the properties stated in Theorem 5.2,

(2) Multikey FHE with distributed setup and split decryption $\Psi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Aux}, \mathsf{Hint}, \mathsf{HRec})$, and

(3) Cryptographic pseudorandom generator $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{\mathsf{poly}(\lambda)}$.

Our compiler is described below.

**First Message:** On input the security parameter $1^\lambda$, the input $x$, and the random coins $s$, the first message algorithm samples (deterministically from $s$) a uniform $\mathsf{seed}$ and a block of the public parameters $\mathsf{pp}_i \leftarrow \mathsf{Setup}(1^\lambda, i)$. Then it returns

$$(\mathsf{pp}_i, \mathsf{First}_i(1^\lambda, (\mathsf{seed}, x), s)).$$

**Second Message:** On input the security parameter $1^\lambda$, an input $x$, the random coins $s$, a function $f$, and the first messages $((\mathsf{pp}_1, y_1), \ldots, (\mathsf{pp}_N, y_N))$ the second message algorithm sets $\mathsf{pp} = \mathsf{pp}_1 \| \ldots \| \mathsf{pp}_N$ and recomputes $\mathsf{seed}$ from $s$. The algorithm sets $(r^{(k)}, r^{(e)}, r^{(a)}) \leftarrow \mathsf{PRG}(\mathsf{seed})$ and computes

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}; r^{(k)}) \text{ and } c \leftarrow \mathsf{Enc}(\mathsf{pk}, x; r^{(e)})$$

$$\text{and } (\alpha, \mathsf{st}) \leftarrow \mathsf{Aux}(\mathsf{pk}, i; r^{(a)}).$$

Let $g((\mathsf{seed}_1, x_1), \ldots, (\mathsf{seed}_N, x_N))$ be defined as follows:

- For all $i \in \{1, \ldots, N\}$ compute:
  - $(r_i^{(k)}, r_i^{(e)}, r_i^{(a)}) \leftarrow \mathsf{PRG}(\mathsf{seed}_i)$
  - $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}; r_i^{(k)})$
  - $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i; r_i^{(e)})$

– $(\alpha_i, \mathsf{st}_i) \leftarrow \mathsf{Aux}(\mathsf{pk}_i, i; r_i^{(a)})$

- Compute $\tilde{c} \leftarrow \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), f, (c_1, \ldots, c_N))$
- Return $\mathsf{Hint}(\mathsf{sk}_1, \tilde{c}, \mathsf{st}_1, 1) \| \ldots \| \mathsf{Hint}(\mathsf{sk}_N, \tilde{c}, \mathsf{st}_N, N)$

The second message algorithm returns

$$(f, \mathsf{pk}, c, \alpha, \mathsf{Second}_i(1^\lambda, (\mathsf{seed}, x), s, g, (y_1, \ldots, y_N)))$$

**Reconstruction:** On input the security parameter $1^\lambda$ and the second round messages $((f, \mathsf{pk}_1, c_1, \alpha_1, z_1), \ldots, (f, \mathsf{pk}_N, c_N, \alpha_N, z_N))$, the output reconstruction algorithm runs $(h_1, \ldots, h_N) \leftarrow \mathsf{Output}(1^\lambda, (z_1, \ldots, z_N))$ and recomputes

$$\tilde{c} \leftarrow \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), f, (c_1, \ldots, c_N)).$$

Then it returns $\mathsf{HRec}(\tilde{c}, (h_1, \ldots, h_N), (\alpha_1, \ldots, \alpha_N))$.

To see why the the complexity of the first message of the resulting MPC is independent of the size of the output, observe that each first message consists of two components: (i) the public parameters $\mathsf{pp}_i$ of the distributed setup of the underlying multikey FHE and (ii) the first message of the underlying MPC. For an appropriate instantiation of multikey FHE with distributed setup (such as the one presented in Section 3), the size of $\mathsf{pp}_i$ is bounded by a polynomial $\mathsf{poly}(\lambda, N)$ and in particular is completely independent of the size of the circuit being evaluated. To analyze the complexity of the first message of the underlying MPC observe that the output of the function $g$ as defined above consists of $N$ *short* decryption hints, each of size at most $\mathsf{poly}(\lambda)$, and in particular independent of the output size of $f$. It follows that the size of the first message cannot depend on the output size of $f$.

One caveat that is introduced by our transformation is that it might potentially affect the efficiency of the resulting MPC: Even if we start with an MPC with efficient output reconstruction (independent of the complexity of computing $f$), the output reconstruction algorithm of the resulting output-independent MPC has to recompute (homomorphically) the function $f$. We will discuss later how to generically bypass this shortcoming. In the following we show that the compiler preserves all of the properties of the underlying MPC. In the following analysis we assume without loss of generality that the the adversary always corrupts a maximally large subset of exactly $N-1$ parties. Such an MPC can be then generically transformed to be secure against an arbitrary subset of corrupted users [MW16] without affecting the efficiency nor the assumptions.

**Theorem 5.3** (Output Independent MPC)**.** *Let $\Pi$ be a two-round semi-malicious MPC satisfying the properties stated in Theorem 5.2, let $\mathsf{PRG}$ be a cryptographic pseudorandom generator, and let $\Psi$ be a split multikey FHE with distributed setup.*

*Then our scheme as described above is a 2-round semi-malicious MPC satisfying the following property:*

- *The first round messages are computed using two algorithms $\mathsf{Reuse.First}$ and $\mathsf{NonReuse.First}$ such that the time to compute $\mathsf{Reuse.First}$ is a polynomial in $(\lambda, \ell_{\mathsf{in}})$ and the time to compute $\mathsf{NonReuse.First}$ is a polynomial in $\lambda$. In particular, the complexity of the first round messages is completely independent of the output length of $f$. As before, the first round*

*messages computed by the $i^{th}$ party is the concatenation of the outputs of* $\mathsf{Reuse.First}_i$ *and* $\mathsf{NonReuse.First}_i$*.*

- *Moreover, the algorithm* Reuse.First *is reusable across multiple executions of secure MPC: that is, in the offline phase, every party on input $x_i$ can compute* Reuse.First *and broadcast its output. In the online phase, executed polynomially many times, all the parties securely compute an $N$-party functionality on the same inputs $(x_1, \ldots, x_N)$ with the only difference being that the first round messages are computed using only* NonReuse.First*; in particular the output of* Reuse.First *is reused for every execution of the online phase.*

*Proof.* We first argue about correctness. By the correctness of the underlying MPC we have that

$$h_i = \mathsf{Hint}(\mathsf{sk}_i, \tilde{c}, \mathsf{st}_i, i)$$

for all $i \in \{1, \ldots, N\}$, where $\tilde{c} = \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), f, (c_1, \ldots, c_N))$ and $c_i = \mathsf{Enc}(\mathsf{pk}_i, m_i)$. Therefore we have that

$$\mathsf{HRec}(\tilde{c}, (h_1, \ldots, h_N), (\alpha_1, \ldots, \alpha_N)) = f(m_1, \ldots, m_N)$$

for honestly computed auxiliary informtion, by the correctness of the split multikey FHE. To show (semi-malicious) security we consider the following sequence of hybrid distributions.

*Hybrid $\mathcal{H}_0$:* This is the distribution induced by the real protocol.

*Hybrid $\mathcal{H}_1$:* In this distribution the messages of $\Pi$ are computed using the corresponding simulator. By the semi-malicious security of $\Pi$, the resulting distribution is computationally indistinguishable from that of the previous hybrid.

*Hybrid $\mathcal{H}_2$:* Here the strings $(r_i^{(k)}, r_i^{(e)}, r_i^{(a)})$, where the $i$-th party is the honest one, are computed using truly random coins. Indistinguishability follows from the pseudorandomness of PRG.

*Hybrid $\mathcal{H}_3$:* This distribution is identical to the previous one except that the decryption hint of the honest party is computed using the simulator of the multikey FHE, instead of using the algorithm Hint. The distribution induced by this hybrid is statistically close to the previous one.

*Hybrid $\mathcal{H}_4$:* Here $c_i$ is computed as $\mathsf{Enc}(\mathsf{pk}_i, 0)$, where 0 is the 0-string padded to the appropriate length. Indistinguishability follows from the semantic security of the multikey FHE with distributed setup. Note that at this point any information about the input of the honest party is lost and we can define the simulator to behave exactly as in this hybrid. This concludes our proof. □

### 5.3 Semi-Malicious Security Preserving Reusability Transformation

In the final step, we show how to transform a two-round semi-malicious MPC with non-reusable first message but satisfying input independence and output independence (as guaranteed by Theorem 5.3) into a two-round semi-malicious MPC with reusable first round message.

To achieve, our starting point is the theorem by [AJJ20] who show the following. We include a proof sketch of the theorem in Appendix A.1.

**Theorem 5.4** (Semi-Honest Reusability Compiler [AJJ20])**.** *Let $\Pi$ be a two-round (non-reusable) delayed-function semi-honest secure computation protocol for $f$ with the following property:*

- *The time complexity to compute the first round message is polynomial in $\lambda$, input, output length of $f$.*

- *The output length of the first round message is a fixed polynomial in $\lambda$ and the number of parties.*

*Then there exists a two-round secure computation protocol satisfying reusable semi-honest property (Definition 2.6).*

We modify the above compiler in two ways. Firstly, we weaken the requirement on the time complexity of the first round messages of the underlying non-reusable protocol. The first round message computation $\mathsf{First}$ consists of two algorithms $\mathsf{Reuse.First}$ and $\mathsf{NonReuse.First}$; the runtime complexity of $\mathsf{Reuse.First}$ is a fixed polynomial in $\lambda$ and the runtime complexity of $\mathsf{NonReuse.First}$ can depend on the input length. Secondly, we start with a non-reusable semi-malicious MPC protocol and then we argue that the reusability compiler preserves the semi-malicious security; thus, the resulting MPC protocol satisfies reusable semi-malicious security. Formally, we prove the following theorem. The proof is more or less identical to the proof of Theorem 5.4 and hence, we include a proof sketch in Appendix A.2.

**Theorem 5.5** (Semi-Malicious Reusability Compiler)**.** *Let $\Pi$ be any two-round semi-malicious MPC satisfying the following properties:*

- *The first round messages are computed using two algorithms* $\mathsf{Reuse.First}$ *and* $\mathsf{NonReuse.First}$ *such that the time to compute* $\mathsf{Reuse.First}$ *is a polynomial in $(\lambda, \ell_{\mathsf{in}})$ and the time to compute* $\mathsf{NonReuse.First}$ *is a polynomial in $\lambda$. In particular, the complexity of the first round messages is completely independent of the output length of $f$. As before, the first round messages computed by the $i^{th}$ party is the concatenation of the outputs of* $\mathsf{Reuse.First}_i$ *and* $\mathsf{NonReuse.First}_i$*.*

- *Moreover, the algorithm* $\mathsf{Reuse.First}$ *is reusable across multiple executions of secure MPC: that is, in the offline phase, every party on input $x_i$ can compute* $\mathsf{Reuse.First}$ *and broadcast its output. In the online phase, executed polynomially many times, all the parties securely compute an $N$-party functionality on the same inputs $(x_1, \ldots, x_N)$ with the only difference being that the first round messages are computed using only* $\mathsf{NonReuse.First}$*; in particular the output of* $\mathsf{Reuse.First}$ *is reused for every execution of the online phase.*

*Then there exits a two-round MPC satisfying reusable semi-malicious security (Definition 2.6).*

# 6 Putting Things Together

The multikey FHE scheme $\Psi$ can be built assuming the hardness of the learning with errors over rings (RingLWE) and the decisional small polynomial ratio (DSPR) problem [LTV12]. On the other hand, non-reusable 2-round MPC can be constructed from the minimal assumption of any 2-round oblivious transfer [GS18, BL18]. Since (single key) homomorphic encryption with re-randomizable ciphertexts suffices to build 2-round (semi-honest) oblivious transfer, we can instantiate it from the scheme proposed in [LTV12] (see [CO17] for details). Stated differently, assuming the semantic security of the scheme proposed in [LTV12] suffices to construct *non-reusable* multikey FHE.

Other than being non-reusable, this instantiation has also the drawback that the size of the public keys $\mathsf{pk}$ is bounded by $\mathsf{poly}(\lambda, N, |f(m_1, \ldots, m_N)|)$. I.e. it additionally grows with the size of the output of $f$ and therefore imposes a bound on the output size of the homomorphic computation. However, the notion of non-reusable multikey FHE still suffices to construct (non-reusable) 2-round MPC. This allows us to state the following lemma.

**Lemma 2.** *If the multikey FHE scheme from [LTV12] is semantically secure, then there exists a (non-reusable)* 2*-round semi-malicious MPC where the communication complexity depends polynomially on the size of the inputs, the size of the outputs, the security parameter, the number of parties.*

Given this building block, we can now apply the results of Section 5 to (i) remove the dependency on the size of the inputs and on the size of the outputs and consequently (ii) obtain a *reusable* 2-round MPC by an invocation of Theorem 5.5. I.e. we obtain the following implication.

**Lemma 3.** *If the multikey FHE scheme from [LTV12] is semantically secure and the LWE problem is hard, then there exists a reusable* 2*-round semi-malicious MPC.*

Note that, in order to achieve reusability, we sacrificed both succinct communication complexity and efficient reconstruction (both depend on the size of the circuit representation of the function that we want to evaluate). To obtain our final result, we apply *again* the compiler presented in Section 4. Note that in this context we can allow the computation and communication complexity of the MPC to grow with the circuit size, since it is anyway computing a circuit whose size is *fixed*. We also stress that, in the reusable settings, we can assume without loss of generality that the homomorphically evaluated functions have one bit of output (thus removing the dependency with respect of the output size), since we can compute multiple bits in parallel. This yields our final result.

**Lemma 4.** *If the multikey FHE scheme from [LTV12] is semantically secure and the LWE problem is hard, then there exists a multikey FHE in the plain model.*

## References


[ABJ+19] Prabhanjan Ananth, Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. From FE combiners to secure MPC and back. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 199–228. Springer, Heidelberg, December 2019.

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

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

[AIK11] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 120–129. IEEE Computer Society Press, October 2011.

[AJJ20] Prabhanjan Ananth, Abhishek Jain, and Zhengzhong Jin. Multiparty homomorphic encryption (or: On removing setup in multi-key fhe). Cryptology ePrint Archive, 2020. `https://eprint.iacr.org/`.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication,

computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012.

[AJN+16] Prabhanjan Ananth, Aayush Jain, Moni Naor, Amit Sahai, and Eylon Yogev. Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 491–520. Springer, Heidelberg, August 2016.

[AJS17] Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Robust transforming combiners from indistinguishability obfuscation to functional encryption. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 91–121. Springer, Heidelberg, April / May 2017.

[BDGM19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 407–437. Springer, Heidelberg, December 2019.

[BDGM20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate io from homomorphic encryption schemes. In *EUROCRYPT (to appear)*, 2020.

[BGJ+17] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Dakshita Khurana, and Amit Sahai. Round optimal concurrent MPC via strong simulation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 743–775. Springer, Heidelberg, November 2017.

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

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

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

[CCS19] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Multi-key homomorphic encryption from TFHE. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 446–472. Springer, Heidelberg, December 2019.

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

[CO17] Wutichai Chongchitmate and Rafail Ostrovsky. Circuit-private multi-key FHE. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 241–270. Springer, Heidelberg, March 2017.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 93–122. Springer, Heidelberg, August 2016.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

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

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-Hop homomorphic encryption and rerandomizable Yao circuits. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 155–172. Springer, Heidelberg, August 2010.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377. ACM Press, May 1982.

[Gol09] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

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

[HW15] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015*, pages 163–172. ACM, January 2015.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st ACM STOC*, pages 12–24. ACM Press, May 1989.

[LATV13] Adriana Lopez-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. Cryptology ePrint Archive, Report 2013/094, 2013. `https://eprint.iacr.org/2013/094`.

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

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012.

[MT19] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic time-lock puzzles and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 620–649. Springer, Heidelberg, August 2019.

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

[PRS17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 461–473. ACM Press, June 2017.

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

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th FOCS*, pages 859–870. IEEE Computer Society Press, October 2018.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

# A Reusability Transformation

## A.1 Proof Sketch of Theorem 5.4

We restate theorem 5.4 below.

**Theorem A.1** (Semi-Honest Reusability Compiler [AJJ20])**.** *Let* $\Pi$ *be a two-round (non-reusable) delayed-function semi-honest protocol with the following property:*

- *The time complexity to compute the first round message is polynomial in* $\lambda$*, input, output length of* $f$*.*
- *The output length of the first round message is a fixed polynomial in* $\lambda$ *and the number of parties.*

*Then there exists a two-round protocol satisfying reusable semi-honest property (Definition 2.6).*

*Proof Sketch.* Let us start with a simple example: the goal is to build a two-round semi-honest MPC protocol for a circuit class $\mathcal{C} = \{C_0, C_1\}$ that allows for reusability of the first message for only two different executions of the second round messages. The starting point is a non-reusable protocol satisfying the efficiency requirements defined in the statement of the theorem above.

We employ a tree-based approach to solve this problem. Let $x_i$, for $i \in [N]$, be the input of the $i^{th}$ party.

- The tree associated with this scheme consists of three nodes: a single root and two leaves. The first leaf is associated with the circuit $C_0$ and the second leaf is associated with the circuit $C_1$.

- Denote the non-reusable MPC associated with the root to be $\Pi_\perp$, with the left leaf to be $\Pi_0$ and the right leaf node to be $\Pi_1$.

- Denote $\widetilde{C}_{i,b}$ to be the garbling of a circuit that takes as input $\Pi_b$ first round messages computed on inputs $x_1, \ldots, x_N$ and generates the $i^{th}$ party's second round round message for the delayed-functionality (represented as a circuit) $C$.

- Finally, denote the circuit $C_\perp$ to be the circuit that takes as input $(x_1, \ldots, x_N)$ and produces:
  - wire labels, associated with $\widetilde{C}_{i,0}$ for the $i^{th}$ party's first round message on input $x_i$ in the protocol $\Pi_0$ and,
  - wire labels, associated with $\widetilde{C}_{i,1}$, for the $i^{th}$ party's first round message on input $x_i$ in the protocol $\Pi_1$.

Armed with the above notation, we present an overview of construction of a two-round MPC for $\mathcal{C} = \{C_0, C_1\}$ allowing for two-time reusability of the first round messages:

- The $i^{th}$ party, for $i \in [N]$, on input $x_i$, produces the first round message $\mathsf{firmsg}^i_\perp$ of the protocol $\Pi_\perp$. It broadcasts $\mathsf{firmsg}^i_\perp$.

- Every party then receives the delayed function $C_b$, for $b \in \{0, 1\}$. On input $C_b$, the $i^{th}$ party does the following:
  - It computes the second round message of $\Pi_\perp$ on input circuit $C_\perp$ to obtain $\mathsf{secmsg}^i_\perp$.
  - It computes a garbled circuit $\widetilde{C}_{i,b}$ as described above.

  It then broadcasts $(\mathsf{secmsg}^i_\perp, \widetilde{C}_{i,b})$ to all the parties.

- Finally, to reconstruct the output, perform the following operations:
  - First compute the final reconstruction procedure of $\Pi_\perp$ to obtain the wire labels corresponding to all the garbled circuits $\widetilde{C}_{1,b}, \ldots, \widetilde{C}_{N,b}$.
  - Then evaluate all the garbled circuits to obtain the second round messages of the protocol $\Pi_b$.
  - Using the second round messages of the protocol $\Pi_b$, reconstruct the output $C_b(x_1, \ldots, x_N)$.

**Full-Fledged Tree-Based Approach.** We can generalize the above approach to construct a reusable two-round MPC scheme for any circuit class and that handles any polynomially many queries. If $s$ is the maximum size of the circuit in the class of circuits, we consider a binary tree of depth $\log(s)$.

- Every edge in the tree is labeled. If an edge $e$ is incident from the parent to its left child then label it with 0 and if $e$ is incident from the parent to its right child then label it with 1.

- Every node in the tree is labeled. The label is the concatenation of all the edge labels on the path from the root to the node.
- Every leaf is associated with a circuit of size $s$.

With each node $v$, associate with $v$ a new instantiation of a one-time pMHE scheme, that we denote by $\Pi_{\mathbf{l}(v)}$, where $\mathbf{l}(v)$ is the label associated with node $v$. If $v$ is the root node $\mathbf{l}(v) = \perp$.

Informally, the encryption algorithm of pMHE generates $\Pi_\perp$ encryption of $x_i$ under the $i^{th}$ secret key. During the evaluation procedure, on input $C$, we generate $\log(s)$ garbled circuits, one for every node on the path from the root to the leaf labeled with $C$. The role of these garbled circuits is to delegate the computation of the second round messages to the final decryption phase. In more detail, the garbled circuit associated with the node $v$ computes the second round messages of the protocol $\Pi_{\mathbf{l}(v||0)}$ and $\Pi_{\mathbf{l}(v||1)}$ and outputs the corresponding wire labels for the garbled circuits associated with its children.

During the final decryption, starting from the root node, each garbled circuit (of every party) is evaluated to obtain wire labels of the garbled circuit associated with the child node on the path from the root to the leaf labelled with $C$. Finally, the garbled circuit associated with the leaf labelled with $C$ is then evaluated to obtain the second round messages of $\Pi_C$. The second round messages are then decoded to recover the final output $C(x_1, \ldots, x_N)$. □

## A.2 Proof Sketch of Theorem 5.5

We first recall the state of the theorem 5.5.

**Theorem A.2** ([AJJ20])**.** *Let* $\Pi$ *be a two-round semi-malicious MPC with succinct (non-reusable) first message*

- *The first round messages are computed using two algorithms* $\mathsf{Reuse.First}$ *and* $\mathsf{NonReuse.First}$ *such that the time to compute* $\mathsf{Reuse.First}$ *is a polynomial in* $(\lambda, \ell_{\mathsf{in}})$ *and the time to compute* $\mathsf{NonReuse.First}$ *is a polynomial in* $\lambda$*. In particular, the complexity of the first round messages is completely independent of the output length of* $f$*. As before, the first round messages computed by the* $i^{th}$ *party is the concatenation of the outputs of* $\mathsf{Reuse.First}_i$ *and* $\mathsf{NonReuse.First}_i$*.*
- *Moreover, the algorithm* $\mathsf{Reuse.First}$ *is reusable across multiple executions of secure MPC: that is, in the offline phase, every party on input* $x_i$ *can compute* $\mathsf{Reuse.First}$ *and broadcast its output. In the online phase, executed polynomially many times, all the parties securely compute an* $N$*-party functionality on the same inputs* $(x_1, \ldots, x_N)$ *with the only difference being that the first round messages are computed using only* $\mathsf{NonReuse.First}$*; in particular the output of* $\mathsf{Reuse.First}$ *is reused for every execution of the online phase.*

*Then there exits a two-round MPC satisfying reusable semi-malicious security (Definition 2.6).*

*Proof Sketch.* As in the proof of the previous theorem, we prove the theorem by first considering the following special case; the argument can be easily generalized for the general case. The goal is to build a two-round semi-malicious MPC protocol for a circuit class $\mathcal{C} = \{C_0, C_1\}$ that allows for reusability of the first message for only two different executions of the second round messages. The starting point is a non-reusable semi-malicious protocol satisfying the efficiency requirements defined in the statement of the theorem above.

We employ a tree-based approach to solve this problem. Let $x_i$, for $i \in [N]$, be the input of the $i^{th}$ party.

- The tree associated with this scheme consists of three nodes: a single root and two leaves. The first leaf is associated with the circuit $C_0$ and the second leaf is associated with the circuit $C_1$.

- Denote the non-reusable MPC associated with the root to $\Pi_\perp$, with the left leaf to be $\Pi_0$ and the right leaf node to be $\Pi_1$.

- Denote $\widetilde{C}_{i,b}$ to be the garbling of a circuit that, has hardwired inside it the outputs of $\Pi_\perp.\mathsf{Reuse.First}_i$ for every $i \in [N]$, it takes as input first round messages of $\Pi_b$ computed using $\Pi_b.\mathsf{NonReuse.First}$ on inputs $x_1, \ldots, x_N$ and generates the $i^{th}$ party's second round round message for the delayed-functionality (represented as a circuit) $C$.

- Finally, denote the circuit $C_\perp$ to be the circuit [4] that takes as input $(x_1, \ldots, x_N)$ and produces:
  - wire labels, associated with $\widetilde{C}_{i,0}$ for the $i^{th}$ party's first round message on input $x_i$ computed using only $\Pi_0.\mathsf{NonReuse.First}$ and,
  - wire labels, associated with $\widetilde{C}_{i,1}$, for the $i^{th}$ party's first round message on $x_i$ computed using only $\Pi_0.\mathsf{NonReuse.First}$.

Armed with the above notation, we present an overview of construction of a two-round MPC for $\mathcal{C} = \{C_0, C_1\}$ allowing for two-time reusability of the first round messages:

- The $i^{th}$ party, for $i \in [N]$, on input $x_i$, produces the first round message $\mathsf{firmsg}^i_\perp = (\mathsf{Reuse.firmsg}^i_\perp, \mathsf{NonReuse.firmsg}^i_\perp)$, computed using the algorithms $\mathsf{Reuse.First}_i$ and $\mathsf{NonReuse.First}_i$ of the protocol $\Pi_\perp$. It broadcasts $\mathsf{firmsg}^i_\perp$.

- Every party then receives the delayed function $C_b$, for $b \in \{0, 1\}$. On input $C_b$, the $i^{th}$ party does the following:
  - It computes the second round message of $\Pi_\perp$ on input circuit $C_\perp$ to obtain $\mathsf{secmsg}^i_\perp$.
  - It computes a garbled circuit $\widetilde{C}_{i,b}$ as described above.

  It then broadcasts $(\mathsf{secmsg}^i_\perp, \widetilde{C}_{i,b})$ to all the parties.

- Finally, to reconstruct the output, perform the following operations:
  - First compute the final reconstruction procedure of $\Pi_\perp$ to obtain the wire labels corresponding to all the garbled circuits $\widetilde{C}_{1,b}, \ldots, \widetilde{C}_{N,b}$.
  - Then evaluate all the garbled circuits to obtain the second round messages of the protocol $\Pi_b$.
  - Using the second round messages of the protocol $\Pi_b$, reconstruct the output $C_b(x_1, \ldots, x_N)$.

The above protocol can be easily generalized to obtain poly-time reusability of the first message.

To see why the resulting protocol satisfies semi-malicious security, note that if the adversary sends garbled circuits in the second round using adversarially chosen randomness then this corresponds to generating the second round messages of the underlying non-reusable protocol

[4] We consider the setting where the circuit is randomized; this is without loss of generality since we can assume that the randomness for this circuit is supplied by the parties

using adversarially chosen randomness. But since our underlying non-reusable protocol satisfies semi-malicious security, it follows that the resulting protocol also satisfies semi-malicious security. □