Towards Round-Optimal Secure Multiparty Computations: Multikey FHE without a CRS

Eunkyung Kim[1], Hyang-Sook Lee(✉)[2], and Jeongeun Park[2]

[1] Security Research Team, Samsung SDS
E tower, Seongchongil 56, Seocho-gu, Seoul, 06765, Republic of Korea Country
ek41.kim@samsung.com
[2] Department of Mathematics, Ewha Womans University, Republic of Korea
hsl@ewha.ac.kr, jungeun7430@ewhain.net

Abstract. Multikey fully homomorphic encryption (MFHE) allows homomorphic operations between ciphertexts encrypted under different keys. In applications for secure multiparty computation (MPC) protocols, MFHE can be more advantageous than usual fully homomorphic encryption (FHE) since users do not need to agree with a common public key before the computation when using MFHE. In EUROCRYPT 2016, Mukherjee and Wichs constructed a secure MPC protocol in only two rounds via MFHE which deals with a common random/reference string (CRS) in key generation. After then, Brakerski et al.. replaced the role of CRS with the distributed setup for CRS calculation to form a four round secure MPC protocol. Thus, recent improvements in round complexity of MPC protocols have been made using MFHE.

In this paper, we go further to obtain round-efficient and secure MPC protocols. The underlying MFHE schemes in previous works still involve the common value, CRS, it seems to weaken the power of using MFHE to allow users to independently generate their own keys. Therefore, we resolve the issue by constructing an MFHE scheme without CRS based on LWE assumption, and then we obtain a secure MPC protocol against semi-malicious security in three rounds.

Keywords: Multi key FHE; LWE assumption; Multi party computation; Lattice.

1 Introduction

1.1 Multikey fully homomorphic encryption.

Fully homomorphic encryption (FHE) scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is a public key encryption scheme with the additional algorithm Eval that allows *homomorphic operations* on ciphertexts: for any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, a function f, and two ciphertexts c, c' encrypted with pk, Eval algorithm takes $(\mathsf{pk}, f, \langle c, c' \rangle)$ as input and returns a new ciphertext c^* such that

$$\mathsf{Dec}(\mathsf{sk}, c^*) = f(\mathsf{Dec}(\mathsf{sk}, c), \mathsf{Dec}(\mathsf{sk}, c')).$$

FHE is a very useful cryptographic primitive, and there has been profound progress after the first construction of FHE by Gentry [4]. *Multikey fully homomorphic encryption (MFHE)*, introduced in [8], is part of that progress. MFHE is a generalization of FHE which supports homomorphic operations between ciphertexts encrypted with *different* keys: with abbreviated notation, Eval algorithm of an MFHE scheme takes c and c' encrypted with pk and pk', respectively, and then returns a new ciphertext c^* such that[3]

$$\mathsf{Dec}(\langle \mathsf{sk}, \mathsf{sk}' \rangle, c^*) = f(\mathsf{Dec}(\mathsf{sk}, c), \mathsf{Dec}(\mathsf{sk}', c'))$$

MFHE can be applied to construct *secure multiparty computation (MPC)* protocols, which is our main concern.

[3] Both of secret keys sk and sk' are needed to decrypt the *multikey ciphertext* c^* for the semantic security.

1.2 Secure multiparty computation via MFHE.

Secure multiparty computation (MPC) can be very helpful for those who want to evaluate a function on their personal data in cooperation with untrusted parties. More specifically, suppose that N parties hold the private input $x_1, \cdots, x_N$, respectively, and that they do not believe one another at all but must evaluate a function f. Then secure MPC protocol allows the parties to compute $f(x_1, \cdots, x_N)$ without disclosing their secret inputs to other users. Secure computation was initially studied by Yao [12] in 1982 as secure two-party computation, which later was generalized to the multi-party by Goldreic, Micali and Wigderson [6].

MPC protocols can be realized by MFHE schemes easily: each user encrypts the data x_i with its own public key pk_i, and sends the ciphertext $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)$ to other users. On receiving all the public keys $\mathsf{pk}_1, \cdots, \mathsf{pk}_N$ and all the ciphertexts $c_1, \cdots, c_N$, users run Eval algorithm of MFHE with inputs $(\{\mathsf{pk}_i\}_{i\in[N]}, \{c_i\}_{i\in[N]}, f)$ to obtain a new ciphertext c^* which encrypts the function value $f(x_1, \cdots, x_N)$. These MPC protocols are not only secure by MFHE, but also highly efficient in terms of round complexity: Mukherjee and Wichs [10] constructed an MFHE scheme based on LWE which simplified the scheme of Clear and McGoldrick [3] to obtain a MPC protocol in only two rounds with a common random/reference string (CRS). They also achieved semi-malicious security for their MPC protocol based on LWE assumption, and fully-malicious security with additional NIZK. And then, Brakerski et al. [2] replaced the CRS in their MFHE scheme with a distributed setup for deriving the CRS, and obtained a three round semi-mailiciously secure MPC protocol and a four round fully-maliciously secure MPC protocol.

However, since these protocols are constructed from MFHE scheme associated with the CRS, either a trusted setup in which all parties get access to the same string CRS (see [10]), or a complex setup for generating the CRS that adds one more round in the protocol (see [2]) is needed. This may weaken the power of using MFHE. Therefore, in order to get a secure MPC protocol which is also simple and round-efficient, it is important to construct an MFHE scheme without CRS.

1.3 Previous Work.

Let us briefly review the MFHE scheme by Mukherjee and Wichs [10] with N parties. Given a common random public matrix $\mathbf{B} \in \mathbb{Z}_q^{(n-1)\times m}$ as a CRS (m and n will be specified later), for $i \in [N]$, i-th party P_i generates a key pair $(\mathsf{pk}_i, \mathsf{sk}_i) = (\mathbf{A}_i, \mathbf{t}_i)$ where $\mathbf{A}_i = (\mathbf{B}, \mathbf{b}_i)^T \in \mathbb{Z}_q^{n\times m}$, $\mathbf{t}_i \in \mathbb{Z}_q^n$ and $\mathbf{t}_i\mathbf{A}_i \approx_q \mathbf{0}$ (i.e. $\mathbf{t}_i\mathbf{A}_i - \mathbf{0}$ is short in $\mathbb{Z}_q^m$). Define the *multi-secret key* $\hat{\mathbf{t}} = (\mathbf{t}_1, \cdots, \mathbf{t}_N) \in \mathbb{Z}_q^{nN}$ which is required for the semantic security. Then a valid *multikey ciphertext* of a bit $\mu \in \{0, 1\}$, which requires all the secret keys $\mathsf{sk}_1, \cdots, \mathsf{sk}_N$ to decrypt, is a matrix $\hat{\mathbf{C}}_i \in \mathbb{Z}_q^{nN\times mN}$ such that $\hat{\mathbf{t}}\hat{\mathbf{C}}_i \approx_q \mu\hat{\mathbf{t}}\hat{\mathbf{G}}$ (i.e. $\hat{\mathbf{t}}\hat{\mathbf{C}}_i - \mu\hat{\mathbf{t}}\hat{\mathbf{G}}$ is short in $\mathbb{Z}_q^{mN}$) where $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ is a fixed public matrix and $\hat{\mathbf{G}} = diag(\mathbf{G}, \cdots, \mathbf{G}) \in \mathbb{Z}_q^{nN\times mN}$ is an expanded matrix having the matrix $\mathbf{G}$ as diagonal components. To do this, they built a polynomial time algorithm $\mathsf{GSW.Lcomb}$ (see Property 5.3 in [10]) that links $\mathsf{pk}_i = \mathbf{A}_i$ and $\mathsf{sk}_j = \mathbf{t}_j$ for $i \neq j$ which is possible thanks to the CRS matrix $\mathbf{B}$. Then the multikey ciphertext $\hat{\mathbf{C}}_i$ is obtained from a *single-key ciphertext* $\mathbf{C}_i$, which can be decrypted by all parties' secret keys. Then they use the MFHE scheme to construct a two round MPC protocol which is secure in the fully-malicious model. See [10] for details.

1.4 Our contribution.

In this work, we give an important stepping stone to get a simple and round-efficient MPC protocol. Namely, we construct a three round MPC protocol, that is secure in the semi-malicious model, without a CRS from an MFHE scheme that use neither a CRS nor a complex setup for inducing a CRS. This is interesting mainly for two reasons. (i) A MPC protocol without a CRS means that no longer a trusted setup (for example, banks, or any certificate authorities) for distributing the CRS is needed, and this fits the recent trends in cryptography such as the famous digital currency Bitcoin. (ii) Three-round seems to be a lower bound when we do not use a CRS: Firstly, since there is no CRS, each user generates its own key pair independently and sends it to other users prior to the protocol, which requires at least one round. Next, once the ciphertexts and public keys are transferred, the computation can be done by the evaluation algorithm of MFHE. Thus, it takes at

least one more round to transfer the information. Finally, since the decryption algorithm of MFHE requires all the secret keys $(\mathsf{sk}_1, \cdots, \mathsf{sk}_N)$ as input due to the semantic security, at least one more round is needed in order for each user to send an intermediate decrypted value involving only its secret key to another users.

To do this, we generalize the MFHE scheme by Mukherjee and Wichs [10] to construct an MFHE scheme without a CRS. In our scheme, P_i freely generates its key pair $(\mathsf{pk}_i, \mathsf{sk}_i) = (\mathbf{A}_i, \mathbf{t}_i)$ by choosing its own random matrix $\mathbf{B}_i \in \mathbb{Z}_q^{(n-1)\times m}$, instead of the CRS matrix $\mathbf{B}$. Namely, we have $\mathsf{pk}_i = \mathbf{A}_i = (\mathbf{B}_i, \mathbf{b}_i)^T \in \mathbb{Z}_q^{n\times m}$. Since pk_i's no longer contain the common matrix $\mathbf{B}$, we cannot apply GSW.Lcomb algorithm directly to link $\mathsf{pk}_i = \mathbf{A}_i$ and $\mathsf{sk}_j = \mathbf{t}_j$ for $i \neq j$. Instead, we give a polynomial time algorithm LinkAlgo that generalizes GSW.Lcomb algorithm. Then we use LinkAlgo algorithm to transform a single-key ciphertext $\mathbf{C}_i$ into a multikey ciphertext $\hat{\mathbf{C}}_i$ as in [10]. Moreover, Since our single key encryption step is independent of the LinkAlgo algorithm, one can use our scheme for single key FHE and then just expand it freely with multi parties if she wants to use it for MFHE or MPC.

We extend the previous version(an MFHE scheme) which was a proceeding paper [7]. Here, we modify the previous scheme a bit(ciphertext expansion procedure) to be more secure adding a new security notion. It is reasonable to think that a multikey cihpertext cannot be decrypted unless all the secret keys are gathered, so we introduce a new security notion called multikey-CPA security and prove the security of our scheme. To do this, we set security parameters of our scheme to satisfy GSW FHE scheme and Regev-LWE symmetric key encryption scheme at the same time. We prove the security by describing a multikey-IND-CPA game to show that the advantage of adversary who distinguishes our MFHE multikey ciphertext is as same as the advantage of the distinguisher of Regev-LWE ciphertext. Since Regev-LWE is semantically secure under certain parameters based on LWE assumption, there is no PPT adversary who distinguishes our multikey ciphertext.

1.5 Organization.

In Section 2, we introduce notation used throughout the paper, and review important definitions, including the learning with errors (LWE) problem and Multikey fully homomorphic encryption (MFHE) schemes. In Section 3, as our first main result, we present LinkAlgo algorithm for transforming a single-key ciphertext to the related multikey ciphertext. Based on the first result, in Section 4, we construct an MFHE scheme without a CRS, and obtain a three round MPC protocol that is secure in the semi-malicious model.

2 Preliminaries

In this paper, we denote κ the *security parameter*. A function $\mathsf{negl}(\kappa)$ is negligible if for every positive polynomial $p(\kappa)$ it holds that $\mathsf{negl}(\kappa) < \frac{1}{p(\kappa)}$. We denote $\mathbb{Z}/q\mathbb{Z}$ as $\mathbb{Z}_q$ and its elements are integer in the range of $(-q/2, q/2]$. Now we define the notation of vectors and matrices. For a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{Z}^n$, $\mathbf{x}$[i] denotes the i-th component scalar. For a matrix $\mathbf{M} \in \mathbb{Z}^{n\times m}$, $\mathbf{M}[i, j]$ denotes the i-th row and the j-th column element of $\mathbf{M}$. Also we use the notation $\mathbf{M}_i^{row}$ which is denoted as i-th row of $\mathbf{M}$ and similarly, $\mathbf{M}_j^{col}$ is denoted as j-th column of $\mathbf{M}$. We use row representation of matrices and define the infinity norm of a vector $\mathbf{x}$ as $\|\mathbf{x}\|_\infty = max_i(\mathbf{x}[i])$ and that of a matrix $\mathbf{M}$ is defined as $max_i(\sum_j \mathbf{M}[i, j])$. Dot product of two vectors $\mathbf{v}, \mathbf{w}$ is denoted by $< \mathbf{v}, \mathbf{w} >$. We also denote the set $\{1, \ldots, n\}$ by $[n]$.

Let X and Y be two distributions over a finite domain. We write $X \overset{comp}{\approx} Y$ if they are computationally indistinguishable. For an integer bound $B_\chi = B_\chi(\kappa)$, we say that a distribution ensemble $\chi = \chi(\kappa)$ is B_χ-bounded if $Pr_{x\leftarrow\chi(\kappa)}[|x| > B_\chi(\kappa)] \leq \mathsf{negl}(\kappa)$. Throughout this paper, we use the notation $\approx_q$ to emphasize that the two values are almost equal in $\mathbb{Z}_q$ except for *short* differences.

2.1 The Learning With Errors Problem.

We recall the learning with errors(LWE) problem, a representative hard problem on lattices introduced by Regev [11]

Definition 1. *Let κ be the security parameter, $n = n(\kappa), q = q(\kappa)$ be integers and let $\chi = \chi(\kappa)$, be distributions over $\mathbb{Z}$. Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and a vector $\mathbf{b} \in \mathbb{Z}_q^n$, the decisional learning with error(LWE) problem is determining whether $\mathbf{b}$ has been sampled uniformly at random from $\mathbb{Z}_q^n$ or $\mathbf{b} = \mathbf{sA} + \mathbf{e}$ for some small random $\mathbf{s} \in \mathbb{Z}_q^m$ and $\mathbf{e} \in \chi^n$ for any polynomial $m = m(\kappa)$.*

The parameter setting for our version of the LWE assumption is that for any polynomial $p = p(\kappa)$ there is a polynomial $n = n(\kappa)$, a modulus $q = q(\kappa)$ of singly-exponential size, and a B_χ bounded distribution $\chi = \chi(\kappa)$ and $q \geq 2^p B_\chi$.

2.2 Multikey FHE(MFHE).

We give a formal definition of Multikey FHE(MFHE) [10] which is an adaptation from the original concept [8].

Definition 2. *A multikey (Leveled) FHE scheme is a tuple of algorithms* $\mathsf{MFHE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Expand}, \mathsf{Eval}, \mathsf{Dec})$ *described as follows.*

- $\mathsf{Setup}(1^\kappa, 1^d) \to \mathsf{params}$*: It takes κ is a security parameter and d is the circuit depth as inputs and it outputs the system parameters* params*.*
- $\mathsf{KeyGen}(\mathsf{params}) \to (\mathsf{pk}, \mathsf{sk})$*: It takes* params *and outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathsf{Enc}(\mathsf{pk}, \mu) \to c$*: On input* pk *and a message μ, outputs a ciphertext c. we call it by a fresh ciphertext.*
- $\mathsf{Expand}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), c, i) \to \hat{c}_i$*: Given a sequence of N public-keys, and a fresh ciphertext c under i-th key* pk_i*, it outputs an expanded ciphertext $\hat{c}$.*
- $\mathsf{Eval}(\mathsf{params}, \mathcal{C}, (\hat{c}_1, \ldots, \hat{c}_\ell)) \to \hat{c}$*: Given a boolean circuit $\mathcal{C}$ of depth $\leq d$ along with ℓ expanded ciphertexts, it outputs an evaluated ciphertext $\hat{c}$.*
- $\mathsf{Dec}(\mathsf{params}, \hat{c}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_N)) \to \mu$*: On input a ciphertext(possibly evaluated) $\hat{c}$ and a sequence of N secret keys, it outputs the message μ. This decryption procedure can be done by the one round threshold distributed decryption:*
  - $\mathsf{PartDec}(\hat{c}, i, \mathsf{sk}_i)$*: On input a ciphertext(possibly evaluated) under a sequence of N public keys and i-th secret key, it outputs a partial decryption p_i.*
  - $\mathsf{FinDec}(p_1, \ldots, p_N)$*: On input N partial decryptions, it outputs the message μ.*

2.3 GSW FHE scheme

Our MFHE scheme is similar to [10] apart from the existence of a trusted setup and a few algorithms. Here we describe the GSW fully homomorphic encryption scheme [5] following the notation of [10]. Note that we take the matrix $\mathbf{B}$ in KeyGen as with the original GSW encryption scheme instead Mukherjee and Wichs [10] gets the matrix $\mathbf{B}$ from Setup, hence consider it as a CRS.

- $\mathsf{GSW.Setup}(1^\kappa, 1^d) \to (\mathsf{params})$: The needed parameters for this scheme to satisfy the LWE assumption are $n, m, q, \mathbf{G}, \chi$ where $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ is a trapdoor matrix [9], B_χ-bounded error distribution $\chi = \chi(\kappa, d)$, a modulus $\mathrm{q} = B_\chi 2^{\omega(d\kappa log\kappa)}$, and $m = n \log q + \omega(\log(\kappa))$. and It outputs $\mathsf{params} := (n, m, q, \mathbf{G}, \chi, B_\chi)$.
- $\mathsf{GSW.KeyGen}(\mathsf{params}) \to (\mathsf{pk}, \mathsf{sk})$: generates a secret key and the corresponding public key respectively. Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^{n-1}$. A secret key $\mathsf{sk} = \mathbf{t} := (-\mathbf{s}, 1) \in \mathbb{Z}_q^n$. Sample $\mathbf{e} \xleftarrow{\$} \chi^m$ and $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{(n-1) \times m}$. Set $\mathbf{b} = \mathbf{sB} + \mathbf{e} \in \mathbb{Z}_q^m$. The corresponding $\mathsf{pk} = \mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is defined as $\mathbf{A} := \begin{pmatrix} \mathbf{B} \\ \mathbf{b} \end{pmatrix}$.
  - The important relation between pk and sk is $\mathbf{tA} \approx_q 0$, which is because $\mathbf{tA} = (-\mathbf{s}, 1) \begin{pmatrix} \mathbf{B} \\ \mathbf{b} \end{pmatrix} = -\mathbf{sB} + \mathbf{b} = \mathbf{e}$: small(i.e.$\|\mathbf{e}\|_\infty \leq B_\chi$).
- $\mathsf{GSW.Enc}(\mathsf{pk}, \mu) \to (\mathbf{C})$: Choose a short random matrix $\mathbf{R} \xleftarrow{\$} \{0,1\}^{m \times m}$ then encrypt a bit message $\mu \in \{0, 1\}$ under the public key pk as $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$, where

$$\mathbf{C} := \mathbf{AR} + \mu \mathbf{G}$$

Here, $\mathbf{tC} = \mathbf{e}' + \mu \mathbf{tG}$ where $\mathbf{e}' = \mathbf{eR}$ implies $\|\mathbf{e}'\|_\infty \leq m B_\chi$.

– $\mathsf{GSW.Eval}(\mathbf{C}_1, \mathbf{C}_2) \rightarrow (\mathbf{C}^*)$: Let $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{Z}_q^{n\times m}$ be two GSW encryption of μ_1, μ_2 under the pk respectively, so that: $\mathbf{tC}_1 = \mu_1\mathbf{tG} + \mathbf{e}_1$ and $\mathbf{tC}_2 = \mu_1\mathbf{tG} + \mathbf{e}_2$. We can do homomorphic operations(addition, multiplication) as following:
  - $\mathsf{GSW.Add}(\mathbf{C}_1, \mathbf{C}_2)$: $\mathbf{C}_1 + \mathbf{C}_2$.
  - $\mathsf{GSW.Mult}(\mathbf{C}_1, \mathbf{C}_2)$: $\mathbf{C}_1\mathbf{G}^{-1}(\mathbf{C}_2) \in \mathbb{Z}_q^{n\times m}$.
– $\mathsf{GSW.Dec}(\mathsf{sk}, \mathbf{C}) \rightarrow (\mu)$: On input as sk, $\mathbf{C}$, set $\mathbf{w} = (0, \ldots, 0, \lfloor q/2 \rceil) \in \mathbb{Z}_q^n$ and compute $\mathbf{v} = \mathbf{tCG}^{-1}(\mathbf{w}^T) = \bar{\mathbf{e}} + \mu(q/2) \in \mathbb{Z}_q$ such that $\bar{\mathbf{e}} =< \mathbf{e}, \mathbf{G}^{-1}(\mathbf{w}^T) >$. Output $|\lfloor \frac{\mathbf{v}}{q/2} \rceil|$ checking if the value is close to 0 or q/2.

The function $\mathbf{G}^{-1}(\cdot)$ introduced in [9] takes any matrix $\mathbf{M} \in \mathbb{Z}_q^{n\times m'}$ (for any $m' \in \mathbb{N}$) and outputs a matrix whose all elements are in the set {0,1}. This function satisfies $\mathbf{GG}^{-1}(\mathbf{M}) = \mathbf{M}$.

The semantic security of GSW FHE scheme under the LWE assumption (with proper parameters) is proved in [5]. To analyze the correctness, we follow the notion of β-noisy ciphertext [10].

Definition 3. *A β-noisy ciphertext of a message μ under a secret key $\mathsf{sk}(= \mathbf{t}) \in \mathbb{Z}_q^n$ is a matrix $\mathbf{C} \in \mathbb{Z}_q^{n\times m}$ satisfying $\mathbf{tC} = \mu\mathbf{tG} + \mathbf{e}$ for some $\mathbf{e}$ with $\|\mathbf{e}\|_\infty \leq \beta$.*

To recover the original message correctly, the maximum size of the error generated during the decryption procedure should be less than $q/4$. Recall that the depth of the circuit is d and let the fresh ciphertext is β-noisy ciphertext. Then β is mB_χ. And evaluated ciphertext is at most $(m+1)^d\beta$-noisy. Finally during the GSW-decryption procedure, the error is multiplied by m. Therefore, the error would become at most $m^2(m+1)^d B_\chi$, which is less than $q/4$ because of our choice of parameters.

2.4 An application of Regev's LWE : a symmetric key cryptosytem(Regev-LWE)

There is an application of LWE problem [11], which is a symmetric key cryptosystem. It is semantically secure assuming the above LWE assumption. We briefly introduce this scheme for our security proof. The parameters such as $\kappa, d, m, n, B_\chi, q$, and the error distribution χ is as same as GSW FHE scheme's.

– $\mathsf{Setup}(1^\kappa)$ outputs $\mathsf{params}=(m, n, q, \chi, B_\chi)$.
– $\mathsf{KeyGen}(\mathsf{params})$: $\mathsf{sk} = s \in \mathbb{Z}_q^n$.
– $\mathsf{Enc}(\mathsf{sk}, \mu)$: It takes a bit μ, and sk.
  - choose $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$
  - Sets a vector $\mathbf{w} = (\lfloor \frac{q}{2} \rfloor, \ldots, \lfloor \frac{q}{2} \rfloor) \in \mathbb{Z}_q^m$
  - Take a random $error \leftarrow \chi^m$.
  - Compute $\mathbf{b} = s\mathbf{A} + error + \mu\mathbf{w} \in \mathbf{Z}_q^m$.
  - Output $c = (\mathbf{A}, \mathbf{b})$.
– $\mathsf{Dec}(\mathsf{sk}, c)$: The decryption is 0, if $\mathbf{b} - s\mathbf{A}$ is closer to 0^m than $\mathbf{w}$. Otherwise, the decryption is 1.

3 MFHE scheme without a CRS

3.1 Single-key ciphertext to multikey ciphertext

An MFHE scheme allows homomorphic operations between ciphertexts under different keys, but the GSW scheme from the previous section is not enough for such operations. This is due to the fact that there is no relation between two different users' keys. In this section, we present a polynomial time algorithm LinkAlgo that links two different keys by giving a relation between them. And then we will use LinkAlgo to transform a *single-key* GSW *ciphertext* into a *multikey ciphertext*, and finally to obtain an MFHE scheme.

Let $\mathbf{R} \in \{0,1\}^{m\times m}$ be a 0-1 matrix, and $\mathbf{V}^{(s,t)}$ be a β-noisy GSW ciphertext of $\mathbf{R}[s,t]$ (s-th row and t-th column of $\mathbf{R}$) under $(\mathsf{pk}, \mathsf{sk}) = (\mathbf{A}, \mathbf{t})$ for all $s, t \in [m]$. Let $(\mathsf{pk}', \mathsf{sk}')$ be another, or possibly same, GSW key pair. Then LinkAlgo takes pk' and encryptions $V^{(s,t)}$'s, and returns a matrix $\mathbf{X}$ as follows:

Algorithm 1 LinkAlgo algorithm

Input: pk' and $\{\mathbf{V}^{(s,t)}\}_{s,t\in[m]}$
Output: $\mathbf{X} \in \mathbb{Z}_q^{n\times m}$

1. Define $\mathbf{L}_{s,t} \in \mathbb{Z}_q^{n\times m}$ for all $s,t \in [m]$ by

$$\mathbf{L}_{s,t}[a,b] = \begin{cases} \mathbf{A}'[a,s] & \text{if t=b} \\ 0 & \text{otherwise} \end{cases}$$

2. Output $\mathbf{X} = \sum_{s=1}^{m}\sum_{t=1}^{m} \mathbf{V}^{(s,t)}\mathbf{G}^{-1}(\mathbf{L}_{s,t}) \in \mathbb{Z}_q^{n\times m}$.

Proposition 4. *We have* $\mathbf{tX} = \mathbf{tA'R} + \mathbf{e}$*, where* $\|\mathbf{e}\|_\infty \le m^3\beta$.

Proof. Since $\mathbf{V}^{(s,t)}$ is a β-noisy encryption of $\mathbf{R}[s,t]$ under $(\mathsf{pk},\mathsf{sk}) = (\mathbf{A},\mathbf{t})$, we have $\mathbf{tV}^{(s,b)} = \mathbf{R}[s,t]\mathbf{tG} + \mathbf{e}_{s,t}$ for some $\mathbf{e}_{s,t}$ with $\|\mathbf{e}_{s,t}\|_\infty \le \beta$. Hence, it holds that

$$\begin{aligned}
\mathbf{tX} &= \sum_{s,t} \mathbf{tV}^{(s,t)}\mathbf{G}^{-1}(\mathbf{L}_{s,t}) \\
&= \sum_{s,t} (\mathbf{R}[s,t]\mathbf{tG} + \mathbf{e}_{s,t})\mathbf{G}^{-1}(\mathbf{L}_{s,t}) \\
&= \sum_{s,t} (\mathbf{R}[s,t]\mathbf{tL}_{s,t} + \mathbf{e}'_{s,t}) \\
&= \mathbf{t}\sum_{s,t} \mathbf{R}[s,t]\mathbf{L}_{s,t} + \sum_{s,t}^{m} \mathbf{e}'_{s,t},
\end{aligned}$$

where $\mathbf{e}'_{s,t} := \mathbf{e}_{s,t}\mathbf{G}^{-1}(\mathbf{L}_{s,t})$ has a norm $\|\mathbf{e}'_{s,t}\| \le m\beta$.

Now it suffices to show that $\sum_{s=1}^{m}\sum_{t=1}^{m} \mathbf{R}[s,t]\mathbf{L}_{s,t} = \mathbf{A'R}$. Note that $\mathbf{L}_{s,t}$ has s-th column of $\mathbf{A}'$ on the t-th column and 0 elsewhere.

$$\begin{aligned}
\sum_{s=1}^{m}\sum_{t=1}^{m} \mathbf{R}[s,t]\mathbf{L}_{s,t} &= \sum_{t=1}^{m}\sum_{s=1}^{m} \begin{pmatrix} 0 \cdots \mathbf{R}[s,t]\mathbf{A}'[1,s] \cdots 0 \\ \vdots \ddots \mathbf{R}[s,t]\mathbf{A}'[2,s] \cdots 0 \\ \vdots \ \vdots \qquad \vdots \qquad \cdots \vdots \\ 0 \cdots \mathbf{R}[s,t]\mathbf{A}'[n,s] \cdots 0 \end{pmatrix} \\
&= \sum_{t=1}^{m} \begin{pmatrix} 0 \cdots \sum_{s=1}^{m} \mathbf{R}[s,t]\mathbf{A}'[1,s] \cdots 0 \\ \vdots \ddots \sum_{s=1}^{m} \mathbf{R}[s,t]\mathbf{A}'[2,s] \cdots 0 \\ \vdots \ \vdots \qquad \vdots \qquad \cdots \vdots \\ 0 \cdots \sum_{s=1}^{m} \mathbf{R}[s,t]\mathbf{A}'[n,s] \cdots 0 \end{pmatrix} \\
&= \sum_{t=1}^{m} \begin{pmatrix} 0 \cdots < \mathbf{A}'^{row}_1, \mathbf{R}^{col}_t > \cdots 0 \\ \vdots \ddots < \mathbf{A}'^{row}_2, \mathbf{R}^{col}_t > \cdots 0 \\ \vdots \ \vdots \qquad \vdots \qquad \cdots \vdots \\ 0 \cdots < \mathbf{A}'^{row}_m, \mathbf{R}^{col}_t > \cdots 0 \end{pmatrix} = \mathbf{A'R},
\end{aligned}$$

where $\mathbf{A}'^{row}_\ell$ denotes the ℓ-th row of $\mathbf{A}'$ and $\mathbf{R}^{col}_\ell$ denotes the ℓ-th colum of $\mathbf{R}$.

To sum up,

$$\mathbf{tX} = \mathbf{t}\sum_{s,t} \mathbf{R}[s,t]\mathbf{L}_{s,t} + \sum_{s,t}^{m} \mathbf{e}'_{s,t} = \mathbf{tA'R} + \mathbf{e},$$

where $\mathbf{e} := \sum_{s=1}^{m}\sum_{t=1}^{m} \mathbf{e}'_{s,t}$ has norm $\|\mathbf{e}\|_\infty \le m^3\beta$. □

3.2 Our MFHE scheme

Let $\mathbf{G}$ be the matrix and $\mathbf{G}^{-1}(\cdot)$ be the function as we described in Section 2. Following the notation of [10], we expand $\mathbf{G}$ as $\hat{\mathbf{G}}_N = diag(\mathbf{G}, \cdots, \mathbf{G}) \in \mathbb{Z}_q^{nN \times mN}$ and let $\hat{\mathbf{G}_N}^{-1}(\cdot)$ be the corresponding function of $\hat{\mathbf{G}}_N$.

Define a tuple of algorithms
$(\mathsf{MFHE.Setup}, \mathsf{MFHE.KeyGen}, \mathsf{MFHE.Enc}, \mathsf{MFHE.Expand}, \mathsf{MFHE.Eval}, \mathsf{MFHE.Dec})$ as follows:

- $\mathsf{MFHE.Setup}(1^\kappa, 1^d) \to (\mathsf{params})$
  1. Run $\mathsf{GSW.Setup}(1^\kappa, 1^d)$
  2. Output params.
- $\mathsf{MFHE.KeyGen}(\mathsf{params}) \to (\mathsf{pk}, \mathsf{sk})$
  1. Run $\mathsf{GSW.KeyGen}(\mathsf{params})$
  2. Output $(\mathsf{pk}, \mathsf{sk}) = \left(\begin{pmatrix} \mathbf{B} \\ \mathbf{b} \end{pmatrix}, \mathbf{t} \right)$.
- $\mathsf{MFHE.Enc}(\mathsf{pk}, \mu) \to (\mathbf{C})$
  1. Run $\mathsf{GSW.Enc}(\mathsf{pk}, \mu)$.
  2. Output $\mathbf{C}$ (i.e. $\mathbf{C} = \mathbf{AR} + \mu\mathbf{G}$).

  In this step, the party keeps its $\mathbf{R}$ for the next step.
- $\mathsf{MFHE.Expand}((\mathsf{pk}_1, \mathsf{pk}_2, \ldots, \mathsf{pk}_N), i, \mathbf{C}) \to (\hat{\mathbf{C}}_i)$ On other's public keys and a fresh ciphertext $\mathbf{C}$, the execution is following:
  1. - $\{\mathbf{V}_{i,j}^{(s,t)}\}_{s,t\in[m]} \leftarrow \{\mathsf{GSW.Enc}(\mathbf{R}[s,t], \mathsf{pk}_j)\}_{s,t\in[m]}$ for $j \in [N]$.
     - $\{\bar{\mathbf{V}}_{i,j}^{(s,t)}\}_{s,t\in[m]} \leftarrow \{\mathsf{GSW.Enc}(\bar{\mathbf{R}}[s,t], \mathsf{pk}_j)\}_{s,t\in[m]}$ for $j \in [N]\backslash\{i\}$, where $\bar{\mathbf{R}}$ is chosen from $\{0,1\}^{m\times m}$.
  2. Compute
     - $\mathbf{X}_i^j \leftarrow \mathsf{LinkAlgo}(\{\mathbf{V}_{i,j}^{(s,t)}\}_{s,t\in[m]}, \mathsf{pk}_i)$ for $j \in [N]$.
     - $\bar{\mathbf{X}}_i^j \leftarrow \mathsf{LinkAlgo}(\{\bar{\mathbf{V}}_{i,j}^{(s,t)}\}_{s,t\in[m]}, \mathsf{pk}_j)$ for $j \in [N]\backslash\{i\}$.
  3. Choose $\mathbf{Q} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$. Set the matrix $\mathbf{Q}_h \in \mathbb{Z}_q^{n\times m}$ having the first row $\mathbf{t}_i\mathbf{Q} + \bar{\mathbf{e}}_h$ and the rest rows zero, where $\mathbf{t}_i$ is the sk of the party i, $\bar{\mathbf{e}}_h$ is chosen from $\chi^m, \forall h \in [N]\backslash\{i\}$.
  4. Define a matrix $\hat{\mathbf{C}}_i \in \mathbb{Z}_q^{nN\times mN}$ as

$$\hat{\mathbf{C}}_i := \begin{bmatrix} \mathbf{C}_i^1 & 0 & \ldots & 0 & 0 \\ 0 & \mathbf{C}_i^2 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{Q} & \ldots & \mathbf{C}_i^i & \ldots & \mathbf{Q} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \ldots & 0 & \mathbf{C}_i^N \end{bmatrix}$$

     which is concatenated by N^2 number of $n \times m$ sub-matrices. The diagonal sub-matrices of $\hat{\mathbf{C}}_i$ are $\mathbf{C}_i^j = \mathbf{C}_i - \mathbf{X}_i^j + \bar{\mathbf{X}}_i^j - \mathbf{Q}_j$ for j$\in [N]\backslash\{i\}$ and the i-th diagonal sub-matrix is $\mathbf{C}_i - \mathbf{X}_i^i$. Lastly, $\mathbf{Q}$ is on the i-th row and zero matrix $0^{n\times m}$ is elsewhere.
  5. Output $\hat{\mathbf{C}}_i$.
- $\mathsf{MFHE.Eval}(\mathsf{params}, f, \hat{\mathbf{C}}_1, \ldots, \hat{\mathbf{C}}_\ell) \to (\hat{\mathbf{C}}^*)$
  1. Given ℓ expanded ciphertexts, run the GSW homomorphic evaluation algorithm working with the expanded dimension nN, mN and $\hat{\mathbf{G}}_N, \hat{\mathbf{G}^{-1}}_N$.
  2. Output $\hat{\mathbf{C}}^*$.
- $\mathsf{MFHE.Dec}(\mathsf{params}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_N), \hat{\mathbf{C}}_i) \to (\mu)$
  1. Given the sequence of secret keys$(\mathsf{sk}_1 = \mathbf{t}_1, \ldots, \mathsf{sk}_N = \mathbf{t}_N)$ and an expanded ciphertext $\hat{\mathbf{C}}_i$, set a vector $\hat{\mathbf{t}}$:$=[t_1, t_2, \ldots, t_N] \in \mathbb{Z}_q^{nN}$.
  2. Run $\mathsf{GSW.Dec}$ algorithm with $\hat{\mathbf{G}}_N$ and $\hat{\mathbf{G}}_N^{-1}$.
  3. Output μ.

In fact, this $\mathsf{MFHE.Dec}$ can be done by threshold decryption, described in Section 2.

- $\mathsf{MFHE.PartDec}(c, \mathsf{sk}_i) \to (p_i)$:
  1. Given an expanded ciphertext $c = \hat{\mathbf{C}}$ and i-th $\mathsf{sk}_i = \mathbf{t}_i \in \mathbb{Z}_q^n$, break $\hat{\mathbf{C}}$ into N row sub matrices $\hat{\mathbf{C}}_i$ (i.e. $\hat{\mathbf{C}} = (\hat{\mathbf{C}}_1^T, \ldots, \hat{\mathbf{C}}_N^T)$ where $\hat{\mathbf{C}}_i \in \mathbb{Z}^{n \times mN}$.
  2. Fix a vector $\hat{\mathbf{w}} = [0, \ldots, 0, \lceil q/2 \rceil] \in \mathbb{Z}_q^{nN}$.
  3. compute $\gamma_i = \mathbf{t}_i \hat{\mathbf{C}}_i \hat{\mathbf{G}}^{-1}(\hat{\mathbf{w}}^T) \in \mathbb{Z}_q$
  4. Output $p_i = \gamma_i + e_i^{sm}$ where $e_i^{sm} \xleftarrow{\$} [-\mathbf{B}_{smdg}^{dec}, \mathbf{B}_{smdg}^{dec}]$ is small randon noise with $\mathbf{B}_{smdg}^{dec} = 2^{d\kappa log\kappa} B_\chi$.
- $\mathsf{MFHE.FinDec}(p_1, \ldots, p_N) \to (\mu)$:
  1. Given $p_1, \ldots, p_N$, just sum $p = \sum_{i=1}^N p_i$.
  2. Output $\mu = |\lceil \frac{p}{q/2} \rfloor|$.

Note that the above algorithms $\mathsf{MFHE.Setup}$, $\mathsf{MFHE.KeyGen}$, and $\mathsf{MFHE.Enc}$ are just GSW scheme's. Since GSW scheme's evaluation algorithm works only for ciphertexts under the same key, Mukherjee and Wichs's MFHE scheme modified it to fit into multikey setting. Therefore, they put the matrix $\mathbf{B}$ in the *SETUP* stage and consider as a common random string (CRS). Due to the fact, one can assume that every user has the same matrix already from a trusted party to generate its public key. However, we give the users freedom of choosing their public keys at random without depending on a CRS. Our MFHE scheme is secure due to the semantic security of GSW encryption scheme and the correctness of expansion.

Correctness of Expansion. Let $\hat{\mathbf{C}}$ be the multikey ciphertext of a bit μ obtained by i-th user from $\mathsf{MFHE.Expand}$ algorithm:

$$\hat{\mathbf{C}} \leftarrow \mathsf{MFHE.Expand}((\mathsf{pk}_1, \cdots, \mathsf{pk}_N), i, \mathbf{C})$$

where $\mathbf{C}$ is a GSW encryption of μ under $(\mathsf{pk}_i, \mathsf{sk}_i) = (\mathbf{A}_i, \mathbf{t}_i)$ and $\mathbf{R}_i$ is the relevant random matrix. For the multi-secret key $\hat{\mathbf{t}} = [\mathbf{t}_1, \cdots, \mathbf{t}_N]$ and the public matrix $\hat{\mathbf{G}_N}$, if $\hat{\mathbf{C}}$ satisfies the relation $\hat{\mathbf{t}}\hat{\mathbf{C}} \approx_q \mu \hat{\mathbf{t}}\hat{\mathbf{G}_N}$, then we can naturally generalize the arguments of GSW FHE scheme. Namely, we can achieve the correctness of encryption, correctness of evaluation, simulatability of partial decryption, and hence a valid MFHE scheme as in [10].

Recall that for a valid GSW key pair $(\mathsf{pk}, \mathsf{sk}) = (\mathbf{A}, \mathbf{t})$ it holds that $\mathbf{tA} = -\mathbf{sB} + \mathbf{b} = \mathbf{e}$ for some $\|\mathbf{e}\|_\infty \leq B_\chi$. For a valid GSW ciphertext $\mathbf{C}$ of μ under $(\mathsf{pk}, \mathsf{sk}) = (\mathbf{A}, \mathbf{t})$ it holds that $\mathbf{tC} = \mu\mathbf{tG} + \mathbf{e}'$ for some $\|\mathbf{e}'\|_\infty \leq \beta_{init} = mB_\chi$. We also recall that for a valid output $\mathbf{X}$ from $\mathsf{LinkAlgo}(\{\mathbf{V}^{(a,b)}\}_{a,b}, \mathsf{pk}' = \mathbf{A}')$ with respect to a 0-1 matrix $\mathbf{R}$ we have $\mathbf{tX} = \mathbf{tA}'\mathbf{R} + \mathbf{e}''$ for some $\|\mathbf{e}''\|_\infty \leq m^3\beta_{init} = m^4 B_\chi$.

Now, we are ready to prove the correctness of expansion. By the definition, we have

$$\begin{aligned}\hat{\mathbf{t}}\hat{\mathbf{C}} &= [\mathbf{t}_1\mathbf{C}_i^1 + \mathbf{t}_i\mathbf{Q}, \mathbf{t}_2\mathbf{C}_i^2 + \mathbf{t}_i\mathbf{Q}, \cdots, \mathbf{t}_i\mathbf{C}_i^i, \cdots, \mathbf{t}_N\mathbf{C}_i^N + \mathbf{t}_i\mathbf{Q}] \\ &= [\mathbf{t}_1(\mathbf{C}_i - \mathbf{X}_i^1 + \bar{\mathbf{X}}_i^1 - \mathbf{Q}_1) + \mathbf{t}_i\mathbf{Q}, \cdots, \mathbf{t}_i(\mathbf{C}_i - \mathbf{X}_i^i), \cdots, \mathbf{t}_N(\mathbf{C}_i - \mathbf{X}_i^N + \bar{\mathbf{X}}_i^N - \mathbf{Q}_N) + \mathbf{t}_i\mathbf{Q}].\end{aligned}$$

Let's see how the bit message μ is correctly recovered and check the error bound by using the followng properties.

(1) $\mathbf{t}_j\mathbf{C} = \mathbf{t}_j(\mathbf{A}_i\mathbf{R}_i + \mu\mathbf{G}) = \mathbf{t}_j\mathbf{A}_i\mathbf{R}_i + \mu\mathbf{t}_j\mathbf{G}$.
(2) $\mathbf{t}_j\mathbf{X}_i^j = \mathbf{t}_j\mathbf{A}_i\mathbf{R}_i + \mathbf{e}_j''$, where $\|\mathbf{e}_j''\|_\infty \leq m^4 B_\chi$.
(3) $\mathbf{t}_i\mathbf{X}_i^i = \mathbf{t}_i\mathbf{A}_i\mathbf{R}_i + \mathbf{e}_i'' = \tilde{\mathbf{e}}_i$, where $\|\tilde{\mathbf{e}}_i\|_\infty \leq (m^4 + m)B_\chi$.
(4) $\mathbf{t}_j\bar{\mathbf{X}}_i^j = \mathbf{t}_j\mathbf{A}_j\bar{\mathbf{R}} + \bar{\mathbf{e}}_j'' = \tilde{\mathbf{e}}_j$, where $\|\tilde{\mathbf{e}}_j\|_\infty \leq (m^4 + m)B_\chi$.
(5) $\mathbf{t}_j\mathbf{Q}_j = \mathbf{t}_i\mathbf{Q} + \bar{\mathbf{e}}_j$, where $\|\bar{\mathbf{e}}_j\|_\infty \leq B_\chi$.

$$\begin{aligned}\therefore \mathbf{t}_j(\mathbf{C} - \mathbf{X}_i^j + \bar{\mathbf{X}}_i^j - \mathbf{Q}_j) + \mathbf{t}_i\mathbf{Q} &= \mu\mathbf{t}_j\mathbf{G} + \hat{\mathbf{e}}_j, \\ \mathbf{t}_i(\mathbf{C} - \mathbf{X}_i^i) &= \mu\mathbf{t}_i\mathbf{G} + \hat{\mathbf{e}}_i\end{aligned}$$

Therefore, we have $\hat{\mathbf{t}}\hat{\mathbf{C}} = \mu\hat{\mathbf{t}}\hat{\mathbf{G}}_N + \hat{\mathbf{e}}$ where $\hat{\mathbf{e}} = [\hat{\mathbf{e}}_1, \cdots, \hat{\mathbf{e}}_i, \cdots, \hat{\mathbf{e}}_N]$ and $\|\hat{\mathbf{e}}\|_\infty \leq (2m^4 + m + 1)B_\chi$. During the decryption procedure, this error is multiplied by mN. By our choice of the parameter, $mN(2m^4 + m + 1)B_\chi < q/4$.

Multikey ciphertext security One thing that is important for multikey FHE encryption scheme's security is that a multikey ciphertext can be decrypted only when all the parties' secret keys are gathered. In other words, no one can decrypt it with only its own secret key. Therefore, it is not enough to be sure about this MFHE scheme's security only with the single key ciphertext security(semantic security of GSW)and the correctness of expansion. We prove no one can decrypt our multikey ciphertext unless he gets all the parties' secret keys. Even if a party who participates in this MFHE scheme has its own secret key and can do something with this secret key, this multikey ciphertext must not reveal any information about other party's message. Moreover, since we do not publish a single key ciphertext but a multikey ciphertext, it is necessary to define a new security notion for our expanded ciphertext indeed.

For a probabilistic multikey FHE encryption algorithm, we naturally extend the original indistinguishability under chosen plaintext attack (IND-CPA) to any multikey FHE scheme by the following game between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. For any multikey FHE encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Expand}, \mathsf{Eval}, \mathsf{Dec})$, any adversary $\mathcal{A}$, and any value κ for the security parameter:
The multikey-CPA(Chosen Plaintext Attack) indistinguishability game:

1 $\mathcal{C}$ runs $\mathsf{KeyGen}(1^\kappa)$ to generate random two key pairs $(\mathsf{pk}, \mathsf{sk}), (\mathsf{pk}', \mathsf{sk}')$. Then it publishes $\mathsf{pk}, \mathsf{pk}', \mathsf{sk}'$ to the adversary $\mathcal{A}$ and keeps sk in secret.
2 The adversary may perform a polynomially bounded number of encryptions or other operations.
3 The adversary $\mathcal{A}$ is given input 1^κ and oracle access to $\mathsf{Enc}_{pk}(\cdot)$ and $\mathsf{Expand}(\cdot, \mathsf{pk}, \mathsf{pk}')$, and outputs a pair of messages $m_0,\ m_1$ of the same length. Then it gives these to $\mathcal{C}$.
4 $\mathcal{C}$ chooses a random bit $\leftarrow \{0,1\}$ and a public key pk, computes a ciphertext $c_b \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_b)$. Then it expands it to a multikey ciphertext by running Expand algorithm with another public key and sends it to $\mathcal{A}$.
5 The adversary is free to perform any number of additional computations, encryptions or decrypt by sk'. Finally, it outputs a guess for the value of b'. If $b' = b$, $\mathcal{A}$ wins. (In this case, we say $\mathsf{MPKC}^{CPA}_{\mathcal{A},\Pi}(\kappa) = 1$.)

Definition 5. *A multikey FHE encryption scheme Π has indistinguishable encryptions under a chosen-plaintext attack(or is multikey-CPA secure) if for all PPT adversaries $\mathcal{A}$ there exists a negligible function* negl *such that*

$$Pr\left[\mathsf{MPKC}^{CPA}_{\mathcal{A},\Pi}(\kappa) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\kappa).$$

Lemma 6. *Assuming LWE assumption, if there exists a PPT adversary $\mathcal{A}$ who distinguishes between our MFHE multikey encryptions of* 0 *and* 1 *with non-negligible probability, there exists a PPT distinguisher $\mathcal{D}$ which distinguishes between* Regev-LWE *encryptions of* 0 *and* 1.

Proof. We prove this statement by contraposition. The proof is done by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

1 The challenger runs $\mathsf{MFHE}.\mathsf{KeyGen}$ to generate key pairs $(\mathsf{pk}\,1, \mathsf{sk}\,1), (\mathsf{pk}\,2, \mathsf{sk}\,2)$ and sends $\mathsf{pk}\,1, \mathsf{pk}\,2, \mathsf{sk}\,2$ to an adversary $\mathcal{A}$ who wants to distinguish multikey ciphertexts.
2 $\mathcal{A}$ may perform a polynomially bounded number of multikey encryptions or other operation running $\mathsf{MFHE}.\mathsf{Enc}$ and $\mathsf{MFHE}.\mathsf{Expand}$ on input whatever she chooses and given public keys.
3 $\mathcal{A}$ submits two distinct chosen plaintext M_0, M_1 to the challenger $\mathcal{C}$.
4 $\mathcal{C}$ chooses a bit $b \in \{0,1\}$ uniformly at random, and encrypts M_b by $\mathsf{pk}\,1$, say $\mathbf{C}_b$, and expands it with the other public key $\mathsf{pk}\,2$ then sends $\hat{\mathbf{C}}_b$ to $\mathcal{A}$.
5 $\mathcal{A}$ multiplies $\mathsf{sk}\,2$ to the second diagonal matrix of $\hat{\mathbf{C}}_b$ on the left then sends it and $\mathsf{sk}\,2$ to $\mathcal{D}$.
6 $\mathcal{A}$ outputs whatever $\mathcal{D}$ outputs.

In detail, in this game, the expanded ciphertext is following

$$\hat{\mathbf{C}}_b = \begin{pmatrix} \mathbf{C}_b - \mathbf{X}_1^1 & \mathbf{Q} \\ 0 & \mathbf{C}_b - \mathbf{X}_1^2 + \bar{\mathbf{X}}_1^2 - \mathbf{Q}_2 \end{pmatrix} \in \mathbb{Z}_q^{2n \times 2m}$$

. In step 5, $\mathcal{A}$ does $\mathbf{t}_2(\mathbf{C}_b - \mathbf{X}_1^2 + \bar{\mathbf{X}}_1^2 - \mathbf{Q}_2)$ and gets $b\mathbf{t}_2\mathbf{G} + \mathbf{t}_1\mathbf{Q} + error$ since $\mathsf{sk}\,2 = \mathbf{t}_2$ in our scheme. Now it is a Regev-LWE ciphertext of b under $\mathsf{sk}\,1$ (Here, $\mathbf{t}_2\mathbf{G}$ is working as $\mathbf{w}$ in Regev-LWE ciphertext). Then $\mathcal{A}$ sends it and $\mathsf{sk}\,2$ together to the Regev-LWE distinguisher $\mathcal{D}$. Finally The advantage of $\mathcal{A}$ is as same as the advantage of $\mathcal{D}$ which is negligible in the parameter κ by the semantic security of Regev-LWE public key encryption scheme [11]. Let $\mathbf{D}_0$ and $\mathbf{D}_1$ are two distributions of Regev-LWE encryptions of 0 and 1, respectively.

$$|Pr\left[\mathsf{MPKC}^{CPA}_{\mathcal{A},\mathsf{MFHE}}(\kappa) = 1\right] - \frac{1}{2}| = |Pr_{x\leftarrow\mathbf{D}_0}\left[\mathcal{D}(x) = 1\right] - Pr_{x\leftarrow\mathbf{D}_1}\left[\mathcal{D}(x) = 1\right]| \leq \mathsf{negl}(\kappa)$$

Therefore, since there is no PPT distinguisher $\mathcal{D}$ who succeeds with non-negligible probability, neither does $\mathcal{A}$. □

Theorem 7. *Our MFHE scheme defined in this section is multikey-CPA secure under LWE assumption.*

Proof. By the above Lemma 6 and the definition of multikey-CPA security, the proof is done. □

4 A three round MPC protocol

In the previous section, we give the LinkAlgo algorithm to have a relation between two key pairs $(\mathsf{pk}, \mathsf{sk})$ and $(\mathsf{pk}', \mathsf{sk}')$. In this section, we make use of the relation obtained by LinkAlgo algorithm to construct our MFHE scheme, and then we introduce a three round MPC protocol that is secure against semi-malicious adversary from the MFHE scheme. This type of adversary is weaker than standard active malicious adversary but stronger than semi honest adversary who just follows a protocol honestly albeit it wants to know other parties' inputs. We give a definition of *Semi-malicious* adversary model which is introduced in [1].

Semi malicious adversary.

A semi-malicious adversary can corrupt arbitrary number of honest parties. It can deviate a protocol to some extent. In other words, he can choose the randomness of input by himself arbitrarily and adaptively in each round. This choice must explain the message sent by the adversary. It must follow the correct behavior of the honest protocol with inputs and randomness that it knows. We assume that it can be rushing (i.e. after seeing messages from honest parties, it may choose its message.) and also the adversarial parties may abort at any point of the protocol. The proof of the security goes on in the usual way showing that the real model's distribution $\overset{comp}{\approx}$ the ideal one.

4.1 A three round MPC protocol via MFHE

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

Preprocessing. Run $\mathsf{params} \leftarrow \mathsf{MFHE}.\mathsf{Setup}(1^\lambda, 1^d)$. Make sure that all the parties have params.
Input: For $i \in [N]$, each party P_i holds input $x_i \in \{0,1\}$, and wants to compute $f(x_1, \cdots, x_N)$.
Round I. *(Round for public key)* Each party P_i executes the following steps:
- Generate its key pair $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{MFHE}.\mathsf{KeyGen}(\mathsf{params})$.
- Broadcast the public key pk_i.

Round II. *(Round for multikey ciphertext)* Each party P_i for $i \in [N]$ on receiving public keys $\{\mathsf{pk}_k\}_{k\neq i}$ executes the following steps:
- Encrypt the message x_i with its public key pk_i to get a single-key ciphertext $\mathbf{C}_i \leftarrow \mathsf{MFHE}.\mathsf{Enc}(\mathsf{pk}_i, x_i)$. Keep the relevant random matrix $\mathbf{R}_{i,j} \in \{0,1\}^{m\times m}$ to C_i which will be need for $\mathsf{MFHE}.\mathsf{Expand}$.
- Run the expand algorithm to get a multikey ciphertext:

$$\hat{\mathbf{C}}_i \leftarrow \mathsf{MFHE}.\mathsf{Expand}((\mathsf{pk}_1, \cdots, \mathsf{pk}_N), i, \mathbf{C}_i)$$

– Broadcast the multikey ciphertext $\hat{\mathbf{C}}_i$.

Round III. *(Round for partial decryptions)* Each party P_i for $i \in [N]$ on receiving ciphertexts $\{\hat{\mathbf{C}}_k\}_{k\neq i}$ executes the following steps:

– Run the evaluation algorithm to get the evaluated ciphertext:

$$\hat{\mathbf{C}}^* \leftarrow \mathsf{MFHE}.\mathsf{Eval}(f, (\hat{\mathbf{C}}_1, \cdots, \hat{\mathbf{C}}_N))$$

– Run the partial decryption algorithm on $\hat{\mathbf{C}}^*$:

$$p_i \leftarrow \mathsf{MFHE}.\mathsf{PartDec}(\hat{\mathbf{C}}^*, (\mathsf{pk}_1, \cdots, \mathsf{pk}_N), i, \mathsf{sk}_i)$$

– Broadcast the partial decryption p_i of $\hat{\mathbf{C}}^*$.

Output: On receiving all the values $\{p_k\}_{k\neq i}$, run the final decryption algorithm to obtain the function value $f(x_1, \cdots, x_N)$:

$$y \leftarrow \mathsf{MFHE}.\mathsf{FinDec}(p_1, \cdots, p_N),$$

and output $y = f(x_1, \cdots, x_N)$.

Security. The security proof of the above MPC protocol against semi-malicious adversaries is similar to that of the previous work [10]. The proof heavily depends on the simulatability of partial decryption and the multikey-CPA security of our MFHE encryption. By the correctness of expansion in Section 4, our MFHE scheme inherits the simulatability of [10]. They proved the MPC protocol is secure against any static semi-malicious adversaries who corrupt exactly $N-1$ parties at first because of their simulator of the threshold decryption. Then they proved the security against those who corrupt arbitrary number of parties using only pseudorandom functions. We adapt their way apart from the messages of each round, i.e. the simulator's the first round behavior of [10] works in our second round and that of the second round works in our third round.

5 Conclusion

We have presented an MFHE scheme without any CRS based on the LWE assumption, which is more secure than our previous version [7]. As an important application, we have constructed a three round MPC protocol which is secure against semi-malicious adversaries. This seems to be round -optimal among all MPC from MFHE without CRS as we mentioned in introduction. In this work, we also have suggested an important stepping stone to get secure MPC protocol, without any trusted setup, against fully malicious adversaries.

Acknowledgments

We would like to thank Dr. Mehdi Tibouchi for his helpful comment in this work. This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT and Future Planning(Grant Number: 2015R1A2A1A15054564)

References

1. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty computation with low communication, computation and interaction via threshold FHE. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 483–501. Springer, Heidelberg, Germany, Cambridge, UK (Apr 15–19, 2012)
2. Brakerski, Z., Halevi, S., Polychroniadou, A.: Four round secure computation without setup. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017: 15th Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 10677, pp. 645–677. Springer, Heidelberg, Germany, Baltimore, MD, USA (Nov 12–15, 2017)

3. Clear, M., McGoldrick, C.: Multi-identity and multi-key leveled FHE from learning with errors. In: Gennaro, R., Robshaw, M.J.B. (eds.) Advances in Cryptology – CRYPTO 2015, Part II. Lecture Notes in Computer Science, vol. 9216, pp. 630–656. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015)
4. Gentry, C.: A Fully Homomorphic Encryption Scheme. Ph.D. thesis, Stanford, CA, USA (2009), aAI3382729
5. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 75–92. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013)
6. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing. ACM, ACM, New York, NY, USA
7. Kim, E., Lee, H.S., Park, J.: Towards round-optimal secure multiparty computations: Multikey FHE without a CRS. In: Susilo, W., Yang, G. (eds.) ACISP 18: 23rd Australasian Conference on Information Security and Privacy. Lecture Notes in Computer Science, vol. 10946, pp. 101–113. Springer, Heidelberg, Germany, Wollongong, NSW, Australia (Jul 11–13, 2018)
8. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Karloff, H.J., Pitassi, T. (eds.) 44th Annual ACM Symposium on Theory of Computing. pp. 1219–1234. ACM Press, New York, NY, USA (May 19–22, 2012)
9. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 700–718. Springer, Heidelberg, Germany, Cambridge, UK (Apr 15–19, 2012)
10. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 735–763. Springer, Heidelberg, Germany, Vienna, Austria (May 8–12, 2016)
11. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM) 56(6), 34 (2009)
12. Yao, A.C.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science. pp. 160–164. IEEE Computer Society Press, Chicago, Illinois (Nov 3–5, 1982)