CS7111: Topics in Cryptography August 21, 23, 28, 2019

Lectures 7-9: GSW13 FHE Scheme, MW15 Multi-Key FHE Scheme

Lecturer: *Shweta Agrawal* Scribe: *Nikhil Vanjani*

1 Introduction

In the past few lectures we saw the development of FHE, in particular we studied two LWE-based schemes - [BV11] and [Bra12] schemes. In this set of lectures, we will study [GSW13] scheme which improves over the previous schemes in couple of aspects, briefly touch on bootstrapping and study an extension of FHE - Multi-Key FHE wherein multiple users with their own data want to outsource computation over the aggregated data.

Recall that in both the schemes we saw previously, the primary idea was to perform multiplication by tensoring. Doing this introduced two problems - Dimension of ciphertext squares with every multiplication and error size squares at every step. To address the first concern, [BV11] introduced the technique of *Dimension Switching* and to address the second concern, they used the idea of *Bit Decomposition*. Using these, their Leveled-FHE scheme was able to support $O(\log n)$ levels of evaluation depth. [Bra12] used the idea of *Scale Invariance* to support $O(n^\epsilon)$ levels. While this is good, the multiplication is quite non-trivial. Moreover, multiplication in both these schemes need some evaluation keys which - a) contain in some ways the secret key in an encrypted state, as a consequence of which we have to rely on circular security assumption to get Levelled-FHE scheme b) are huge in size (each $\Omega(n^3)$) which incurs a huge computational cost during multiplication. So we would like to see if we can do better? [GSW13] scheme answers these questions in affirmation. They propose a scheme which is conceptually-simpler in that the multiplication is natural and they get rid of evaluation keys altogether, as a consequence, their Levelled-FHE scheme is asymptotically-faster and doesn't rely on circular security assumption. In these lectures, we will also discuss the idea of *bootstrapping* introduced by [Gen09a] [Gen09b] which is the only way till date to convert any Levelled-FHE scheme into a pure-FHE scheme. Performing bootstrapping requires circular security assumption, so GSW's pure-FHE scheme doesn't get rid of it altogether but is a step towards it. **Obtaining pure-FHE scheme without relying on circular security assumption is an open problem as on today.**

2 Gentry, Sahai, Waters 2013 Scheme [GSW13]

2.1 Construction

- $\underline{\mathsf{GSW.Setup}(\mathbf{1^\lambda})}$:
  - Choose $B \leftarrow \mathbb{Z}_q^{(n-1)\times m}$, $s \leftarrow \mathbb{Z}_q^{1\times(n-1)}$, $e \leftarrow \chi^{1\times m}$ where $m = O(n \log q)$
  - Let $b = sB + e \in \mathbb{Z}_q^{1\times m}$
  - Let $t = (-s||1) \in \mathbb{Z}_q^{1\times n}$
  - Let $A = \begin{pmatrix} B \\ b \end{pmatrix} \in \mathbb{Z}_q^{n\times m} \Rightarrow tA = -sB + b = e \approx 0$
  - **Output: $\boldsymbol{pk = A}$, $\boldsymbol{sk = t}$**

- GSW.Enc($pk, \mu \in \{0,1\}$):
  - Choose a short random matrix $R \leftarrow \{0,1\}^{m \times m}$
  - Let $G = \begin{pmatrix} g^T & 0 & 0 & ... & 0 \\ 0 & g^T & 0 & ... & 0 \\ & & . & & \\ & & & . & \\ 0 & ... & 0 & 0 & g^T \end{pmatrix} \in \mathbb{Z}_q^{n \times nk}$
    where $g = (2^0, 2^1, 2^2, ..., 2^{k-1}) \in \mathbb{Z}_q^k$ where $k = \log q$
    - **Property:** $\boldsymbol{G \cdot BitDec(m) = m}$, where m is a vector of size n. Hence, from now on, we will represent $\boldsymbol{BitDec(m)}$ as $\boldsymbol{G^{-1}(m)}$. Note that, here, $\boldsymbol{G^{-1}}$ is just a notation.
  - Output $\boldsymbol{c = AR + \mu G}$
- GSW.Dec(sk, c):
  - Let $w = (0, 0, ..., 0, \lceil q/2 \rceil) \in \mathbb{Z}_q^n$
  - Compute $V = tCG^{-1}(w) = \mu\lceil q/2 \rceil + error$
  - Output $\boldsymbol{Round_{q/2}(V)}$

$$\begin{aligned} tCG^{-1}(w) &= t(AR + \mu G)G^{-1}(w) \\ &= (tAR + \mu tG)G^{-1}(w) \\ &= (eR + \mu tG)G^{-1}(w) \\ &\approx (\mu tG)G^{-1}(w) \\ &= \mu tw \\ &= \mu\lceil q/2 \rceil \end{aligned}$$

- GSW.Eval:
  - **Add**: $\boldsymbol{C_{add} = C_1 + C_2} = A(R_1 + R_2) + (\mu_1 + \mu_2)G \Rightarrow$ Correctness is obvious
  - **Mult**: $\boldsymbol{C_{mult} = C_1 G^{-1}(C_2)}$. This works because -

$$\begin{aligned} tC_{mult} &= (tC_1)G^{-1}(C_2) \\ &\approx (\mu_1 tG)G^{-1}(C_2) \\ &= \mu_1 tC_2 \\ &\approx (\mu_1\mu_2)tG \end{aligned}$$

2.2 Error Analysis

Definition 1. (β-noisy ciphertext). *A β-noisy ciphertext of some message μ encrypted under sk $t \in \mathbb{Z}_q^n$ is a matrix $C \in \mathbb{Z}_q^{n \times m}$ st. $tC = \mu tG + e$, for some e st. $||e||_\infty \leq \beta$*

Addition: If C_1 is β_1-noisy and C_2 is β_2-noisy, then, C_{add} is $(\beta_1 + \beta_2)$-noisy.

Multiplication: $tC_{mult} = e'' + \mu_1\mu_2 tG$, where $e'' = e_1 G^{-1}(C_2) + \mu_1 e_2$
Here, $\mu_1 e_2$ is β_2-noisy as $\mu_1 \in \{0,1\}$ and $||e_1 G^{-1}(C_2)||_\infty \leq \beta_1 m$ as $G^{-1}(C_2) \in \{0,1\}^{m \times m}$.
Therefore, C_{mult} is $(\beta_2 + m\beta_1)$-noisy.

Hence, if we bound $\beta_1, \beta_2 \leq \mathsf{B}$, then noise after 1 level of Eval is bounded by -
$\boldsymbol{\mathsf{B} \cdot O(n \log q) = \mathsf{B} \cdot poly(n)}$ because $m = O(n \log q)$. As seen in previous schemes, with this factor of noise growth, we can support upto $\boldsymbol{D = n^\epsilon}$ levels of Evaluation. Now, let's see how to go from supporting $D = n^\epsilon$ levels to $\boldsymbol{D = poly(n)}$ levels.

2.3 Bootstrapping

Theorem 1. (Bootstrapping Theorem). *If a scheme is capable of evaluating its own decrpytion circuit, it is capable of evaluating any circuit of poly depth.*

Let $\widehat{\boldsymbol{\mu}}$ be a ciphertext corresponding to a plaintext $\boldsymbol{\mu}$ encrypted under secre key $\boldsymbol{sk}$. Then, we have that -

$$\mathsf{FHE.Dec}(\widehat{\mu}, sk) \to \mu$$

where, $\mathsf{FHE.Dec}$ is a circuit with taking 2 input parameters. For the sake of understanding, let's just hardcode the ciphertext in it and call the resulting circuit as $\mathcal{C}$ (This circuit is of depth $\boldsymbol{O(\log n)}$). Then, we have -

$$\mathcal{C}(sk) \to \mu$$

We get that -

$$\mathsf{FHE.Eval}(\mathcal{C}, \widehat{sk} = \mathsf{FHE.Enc}(sk)) \to \mathsf{FHE.Enc}(\mu, sk) = \widehat{\widehat{\mu}}$$

We know that noise after evaluation depends on circuit dpeth and initial noise. Suppose we got $\widehat{\mu}$ in the first place as a result of $d = n^\epsilon$ evaluations. Then, noise in $\widehat{\mu}$ is $\beta_d = \beta_{n^\epsilon}$. Noise in $\widehat{\widehat{\mu}}$ is $\beta_{\log n}$ because depth of circuit $\mathcal{C}$ is $\log n$. Therefore, we are decreasing the noise from β_{n^ϵ} to $\beta_{\log n}$ and hence enabling to perform more evaluations on the ciphertext. This way, we can keep performing evaluations upto unbounded depth. Because, the Eval algo needs to be efficient in run time, we say that we can now perform evaluations upto $\boldsymbol{poly(n)}$ depth.

3 Multi-Key or Multi-user FHE (MFHE)

MFHE enables homomorphic computation over data encrypted under different secret keys. A straightforward application of this scheme as shown by [MW15] is that we can obtain 2-round MPC. They extend the GSW FHE scheme to obtain MFHE and then show how to construct 2-round MPC. In this lecture, we will only see the former and not delve onto the latter as MPC is beyond the scope of this course.

3.1 Multi-Key variant of GSW

For simplicity, let's start with $n = 2$ keys. The keys are sampled just like as in GSW scheme. Note that $\boldsymbol{B}$ is common to both A and A' and we refer to it as the public parameter.

$$sk = t = (-s, 1),\ pk = A = \begin{pmatrix} B \\ b \end{pmatrix}, \text{ where } b = sB + e$$
$$sk' = t' = (-s', 1),\ pk = A' = \begin{pmatrix} B \\ b' \end{pmatrix}, \text{ where } b = s'B + e'$$

Let, $\boldsymbol{C = \mathsf{GSW.Enc}(pk, \mu) = AR + \mu G}$
Therefore, $tC \approx \mu tG$
Consider a hypothetical situation (the **Decryption View**) where we encrypt under $\widehat{t} = (t, t') \in \mathbb{Z}_q^{1\times 2n}$ instead of t -

$$RHS = \mu\widehat{t}\widehat{G} = \mu(t, t') \begin{pmatrix} G & 0 \\ 0 & G \end{pmatrix} = (\mu tG, \mu t'G)$$

$$LHS = \widehat{t}\widehat{C} = (t, t') \begin{pmatrix} C & 0 \\ 0 & C \end{pmatrix} = (tC, t'C) = (eR + \mu tG, (e + (s - s')B)R + \mu t'G)$$

This doesn't work because $(s - s')B$ is not small and consequently we will get some garbage value in the second component. Let's try a new $\widehat{C} \in \mathbb{Z}_q^{2n \times 2m}$ to get this right -

$$\begin{aligned}
LHS &= \widehat{t}\widehat{C} \\
&= (t, t') \begin{pmatrix} C & X \\ 0 & C \end{pmatrix} \\
&= (tC, tX + t'C) \\
&= (eR + \mu tG, tX + (e + (s - s')B)R + \mu t'G) \\
&= (eR + \mu tG, tX + (e + sB - s'B)R + \mu t'G) \\
&= (eR + \mu tG, tX + (e + b - e - b' + e')R + \mu t'G) \\
&= (eR + \mu tG, tX + (e' + b - b')R + \mu t'G) \\
&\approx (\mu tG, tX + (b - b')R + \mu t'G)
\end{aligned}$$

Therefore, we want $\boldsymbol{X}$ st. $\boldsymbol{tX \approx (b' - b)R}$

Lemma 1. *Let $\boldsymbol{M \in \{0,1\}^{m \times m}}$ and $\boldsymbol{C_{i,j}}$ be a $\boldsymbol{\beta}$-noisy* GSW *encryption of $\boldsymbol{M[i,j]}$, where $\boldsymbol{i, j \in [m]}$. Let, $\boldsymbol{t}$ be tje GSW sk. Let $\boldsymbol{v \in \mathbb{Z}_q^{1 \times m}}$ be some vector (not necessarily short). Then, $\exists$ poly. time deterministic algorithm -*

$$C_{lc} = \mathsf{GSW.LComb}(C_{1,1}, ..., C_{m,m}, v) \text{ (LComb stands for Linear Combination)}$$
$$\text{st. } C_{lc} \in \mathbb{Z}_q^{n \times m} \text{ and } tC_{lc} = vM + e, \text{ where } ||e||_\infty \leq m^3\beta$$

GSW.LComb*:*

1. *Define $Z_{i,j} \in Z_q^{n \times m}$ as -*

$$Z_{i,j}[a,b] = \begin{cases} v_i, & \text{if } a = n \text{ and } b = j \\ 0, & \text{otherwise} \end{cases}$$

With this definition, we have -

$$Z_{1,1} = \begin{pmatrix} 0 & . & . & . & . & 0 \\ . & & & & & . \\ . & & & & & . \\ . & & & & & . \\ 0 & . & . & . & . & 0 \\ v_1 & 0 & . & . & . & 0 \end{pmatrix}, \; Z_{1,2} = \begin{pmatrix} 0 & . & . & . & . & 0 \\ . & & & & & . \\ . & & & & & . \\ . & & & & & . \\ 0 & . & . & . & . & 0 \\ 0 & v_1 & 0 & . & . & 0 \end{pmatrix}, \; ...$$

2. *Define C_{lc} as -*

$$C_{lc} = \sum_{i,j \in [m]} C_{i,j} G^{-1}(Z_{i,j})$$

Correctness:

$$\begin{aligned}
tC_{lc} &= \sum_{i,j \in [m]} tC_{i,j} G^{-1}(Z_{i,j}) \\
&= \sum_{i,j \in [m]} (M[i,j]tG + e_{i,j}) G^{-1}(Z_{i,j}) \\
&= t \sum_{i,j \in [m]} M[i,j] Z_{i,j} + error \\
&= t \sum_{i \in [m]} \Big(\sum_{j \in [m]} M[i,j] Z_{i,j} \Big) + error
\end{aligned}$$

$$
\begin{aligned}
&= t \sum_{i \in [m]} \begin{pmatrix} 0 \\ v_i M[i] \end{pmatrix} + error \\
&= t \begin{pmatrix} 0 \\ vM \end{pmatrix} + error \\
&= (-s, 1) \begin{pmatrix} 0 \\ vM \end{pmatrix} + error \\
&= vM + error
\end{aligned}
$$

□

Let's try to use this Lemma to get $\boldsymbol{X}$ st.$\boldsymbol{tX \approx (b' - b)R}$. Informally, let $\boldsymbol{v = b' - b}$, $\boldsymbol{M = R}$, if $\boldsymbol{C_{i,j}}$ be GSW encryption of $\boldsymbol{R_{i,j}}$, then, Lemma 1 gives us $\boldsymbol{X = C_{lc}}$ st. $\boldsymbol{tX \approx (b' - b)R}$. Now that we have obtained a way to $\widehat{C}$ from C (we call this as *Masking Scheme for GSW* which

we will see formally shortly), we can do this for ciphertexts of both parties and then perform homomorphic computation over them. The only issue remaining to address is how to decrpyt the ciphertexts? Let's try this -

$$
\begin{aligned}
Dec(\widehat{t}, \widehat{C}) &= \widehat{t}\widehat{C}\widehat{G^{-1}}(\widehat{w}), \text{ where } \widehat{w} = (0, ..., 0, \lfloor q/2 \rfloor) = (0^n, w) \in \mathbb{Z}_q^{2n} \\
&= \mu \widehat{t}\widehat{G}\widehat{G^{-1}}(\widehat{w}) + error \\
&= \mu(t, t') \begin{pmatrix} G & 0 \\ 0 & G \end{pmatrix} \begin{pmatrix} G^{-1}(0^n) \\ G^{-1}(w) \end{pmatrix} + error \\
&= \mu(tG, t'G) \begin{pmatrix} G^{-1}(0^n) \\ G^{-1}(w) \end{pmatrix} + error \\
&= \mu(t0^n + t'w) + error \\
&= \mu \lfloor q/2 \rfloor + error
\end{aligned}
\tag{1}
$$

In theory and for the sake of the definition in next section, this will work. But clearly, both parties can't have both sk and sk' as it will violate security. So, in a practical scenario we perform threshold decryption which involves both parties getting partial decryption using their decryption keys and then they share the partial decryptions with each other. Using both partial decrpytions (p_1, p_2), they individually obtain the decryption. This is done as follows - For 2 keys, let's see $\widehat{C}$ as consisting of 2 matricies - $\widehat{C^{(1)}}, \widehat{C^{(2)}} \in \mathbb{Z}_q^{n \times 2m}$ stacked on top of each other, ie,

$$
\widehat{C} = \begin{pmatrix} \widehat{C^{(1)}} \\ \widehat{C^{(2)}} \end{pmatrix}
$$

We set p_1 and p_2 as follows -

$$
\begin{aligned}
p_1 &= t\widehat{C^{(1)}}\widehat{G^{-1}}(\widehat{w}) + e_1 \\
p_2 &= t'\widehat{C^{(2)}}\widehat{G^{-1}}(\widehat{w}) + e_2
\end{aligned}
$$

where e_i is some medium-sized smudging error. This error is needed to smudge out any information about the error contained in the ciphertext $\widehat{C}$, which might contain sensitive information beyond just the plaintext bit. We get that -

$$
\begin{aligned}
p_1 &= t(C,X)\begin{pmatrix}G^{-1}(0^n)\\ G^{-1}(w)\end{pmatrix} + e_1\\
&= (tC, tX)\begin{pmatrix}G^{-1}(0^n)\\ G^{-1}(w)\end{pmatrix} + e_1\\
&= tCG^{-1}(0^n) + tXG^{-1}(w) + e_1\\
&\approx \mu tGG^{-1}(0^n) + (b'-b)RG^{-1}(w)\\
&= \mu t 0^n + (b'-b)RG^{-1}(w)\\
&= (b'-b)RG^{-1}(w)\\
p_2 &= t'(0,C)\begin{pmatrix}G^{-1}(0^n)\\ G^{-1}(w)\end{pmatrix} + e_2\\
&= (0, t'C)\begin{pmatrix}G^{-1}(0^n)\\ G^{-1}(w)\end{pmatrix} + e_2\\
&= t'CG^{-1}(w) + e_2\\
&\approx (b-b')RG^{-1}(w) + \mu t'GG^{-1}(w)\\
&= (b-b')RG^{-1}(w) + \mu\lfloor q/2\rfloor
\end{aligned}
$$

The partial decryptions can be combined to obtain the message as follows -

$$p_1 + p_2 \approx (b'-b)RG^{-1}(w) + (b-b')RG^{-1}(w) + \mu\lfloor q/2\rfloor = \mu\lfloor q/2\rfloor$$

Now that we have seen the building blocks, let's see the MFHE scheme formally-

Definition 2. (Multi-Key Levelled FHE scheme). *A multi-key levelled FHE scheme is a tuple of 6 algorithms - (*$\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Expand}, \mathsf{Eval}, \mathsf{Decrypt}$*) described as follows -*

- $\mathsf{Setup}(\mathbf{1^\lambda}, \mathbf{1^d}) \rightarrow \boldsymbol{params}$ *(including matrix B)*
- $\mathsf{KeyGen}(\boldsymbol{params}) \rightarrow \boldsymbol{pk}, \boldsymbol{sk}$
- $\mathsf{Encrypt}(\boldsymbol{pk}, \boldsymbol{\mu}) \rightarrow \boldsymbol{C}$
- $\mathsf{Expand}(\boldsymbol{pk_1}, ..., \boldsymbol{pk_N}, \boldsymbol{i}, \boldsymbol{C}) \rightarrow$ ***expanded ciphertext*** $\widehat{\boldsymbol{C_i}}$*, where input ciphertext C is encrypted under* pk_i
- $\mathsf{Eval}(\mathcal{C}, \widehat{C_1}, ..., \widehat{C_l}, \boldsymbol{params}) \rightarrow \widehat{\widehat{C}}$*, where,* $\mathcal{C}$ *is a boolean circuit of depth* $\leq d$
- $\mathsf{Decrypt}(\boldsymbol{sk_1}, ..., \boldsymbol{sk_N}, \widehat{\widehat{C}}) \rightarrow \boldsymbol{\mu}$

MFHE scheme needs following properties:

- **Correctness**: There are two aspects to it -
  - Correctness of Expand: $\mathsf{Decrypt}(sk_1, ..., sk_N, \widehat{C_i}) = \mu_i$
  - Correctness of Eval: $\mathsf{Decrypt}(sk_1, ..., sk_N, \widehat{\widehat{C}}) = \mathcal{C}(\mu_1, ..., \mu_l)$
- **Security**: For any two messages μ_0, μ_1, the distributions $\boldsymbol{params}, \boldsymbol{pk}, \boldsymbol{Encrypt}(\boldsymbol{pk}, \boldsymbol{\mu_0})$ and
  $\boldsymbol{params}, \boldsymbol{pk}, \boldsymbol{Encrypt}(\boldsymbol{pk}, \boldsymbol{\mu_1})$ are computationally indistinguishable.
- **Compactness**: Size of $\widehat{\widehat{C}}$ should be independent of $\mathcal{C}$ circuit and l.

3.2 Masking Scheme for GSW

- $\underline{\mathsf{UniEnc}(\mu, pk) \to (U, C)}$: where U is some universal info and C is ciphertext
  - $A = pk$
  - $C = \mathsf{GSW.Enc}(pk, \mu) = AR + \mu G$
  - Compute m^2 ciphertexts - $V^{(a,b)} \leftarrow \mathsf{GSW.Enc}(pk, R[a, b])$ for $a, b \in [m]$
  - Set $U = \{V^{(1,1)}, ..., V^{(m,m)}\}$
  - **Output $(\boldsymbol{U}, \boldsymbol{C})$**
- $\underline{\mathsf{Extend}(U, pk, pk') \to X \in \mathbb{Z}_q^{n \times m}}$:
  - Parse $pk = A = \begin{pmatrix} B \\ b \end{pmatrix}$, $pk' = A' = \begin{pmatrix} B \\ b' \end{pmatrix}$, $U = \{V^{(a,b)}\}_{a,b \in [m]}$
  - **Output $\boldsymbol{X} = \mathsf{GSW.LComb}(\boldsymbol{V^{(1,1)}, ..., V^{(m,m)}, b' - b})$**. By guarantees of $\mathsf{GSW.LComb}$ algo, as we saw previously, we get that - $\boldsymbol{tX = (b' - b)R + error}$

Correctness: If U, C, X are chosen as above, then we have that -

$$\boldsymbol{\mu \leftarrow \mathsf{GSW.Dec}(sk, C)} \text{ and } \boldsymbol{tX + t'C = \mu t'G + error}$$

3.3 [MW15] Construction of Multi-Key Levelled FHE

- $\underline{MW.SETUP(1^\lambda 1^D)}$:
  - Choose q, n, $m = O(n \log q)$, $B_\chi$bounded error distribution χ appropriately so that the $LWE_{n-1,m,q,\chi}$ assumption holds. Choose $\boldsymbol{B} \leftarrow \mathbb{Z}_q^{(n-1) \times m}$.
  - **Output** $params = (q, n, m, \chi, B_\chi, \boldsymbol{B})$
- $\underline{\mathsf{MW.KeyGen}(params)}$:
  - For $i \in [N]$, sample $s_i \leftarrow \mathbb{Z}_q^{1 \times (n-1)}, e \leftarrow \chi^{1 \times m}$
  - Let $b_i = s_i B + e_i$
  - Let $t_i = (-s_i || 1) \in \mathbb{Z}_q^{1 \times n}$
  - Let $A_i = \begin{pmatrix} B \\ b_i \end{pmatrix} \in \mathbb{Z}_q^{n \times m}$
  - **Output $\boldsymbol{pk_i = A_i, sk_i = t_i \forall i \in [N]}$**
- $\underline{\mathsf{MW.Encrypt}(pk, \mu)}$:
  - **Output $\boldsymbol{\mathsf{UniEnc}(\mu, pk)}$**
- $\underline{\mathsf{MW.Expand}(pk_1, ..., pk_N, i, C)}$:
  - For $j \in \{pk_1, ..., pk_N\} \backslash \{pk_i\}$, compute - $X_j \leftarrow \mathsf{Extend}(U, pk_i, pk_j)$
  - Define $\widehat{C} \in \mathbb{Z}_q^{nN \times mN}$ as concatenation of N^2 submatrices $C_{a,b}$ where each $C_{a,b} \in \mathbb{Z}_q^{n \times m}$, $a, b \in [N]$ -

$$C_{a,b} = \begin{cases} C & \text{when } a = b \\ X_j & \text{when } a = i \neq j \text{ and } b = j \\ 0^{n \times m} & \text{otherwise} \end{cases}$$

  - **Output $\boldsymbol{\widehat{C}}$**

- MW.Eval:
  - For addition and multiplication use GSW.Add and GSW.Mult respectively but with expanded dimensions - $n' = nN, m' = mN$ and the expanded $\widehat{G}_N, \widehat{G}_N^{-1}$ matricies.
- MW.Decrypt$(sk_1, ..., sk_N, \widehat{\widehat{C}})$:
  - Decryption is done just as described in (1) in section 3.1

3.4 Security of the Scheme

Hybrid 0: $pk = A, C(0, R_0), U(R_0)$

- This hybrid corresponds the honest execution where $\mu_0 = 0$ is encrypted. Note that $C(0, R_0) = AR_0 + (0)G = AR_0$ and $U(R_0) = \{V^{(1,1)}, ..., V^{(m,m)}\}$, where $V^{(a,b)} \leftarrow \mathsf{GSW.Enc}(pk, R_0[a, b])$ for $a, b \in [m]$

Hybrid 1: $pk = A, C(0, R_0), U(0)$

- We modify each $V^{(a,b)}$ to be GSW encryption of 0. So, Hybrid 1 is indistinguishable from Hybrid 0 by semantic security of GSW scheme.

Hybrid 2: $pk = A, C(1, R_1), U(0)$

- We modify the ciphertext C to be encryption of 1 instead of 0. Hence, by semantic security of GSW scheme, Hybrids 1 and 2 are computationally indistinguishable.

Hybrid 3: $pk = A, C(1, R_1), U(R_1)$

- We modify each $V^{(a,b)}$ to be GSW encryption of $R_1[a, b]$. Hence, by semantic security of GSW scheme, Hybrids 2 and 3 are computationally indistinguishable.
- Note that this hybrid corresponds the honest execution where $\mu_0 = 1$ is encrypted and hence is the last hybrid.

MFHE Security Game 1: if $b_1 = 0$, we are in Hybrid 0. if $b_1 = 1$, we are in Hybrid 1

GSW Challenger		**Reduction**		**MFHE Adversary**
$pk = A$	$\longrightarrow$	A	$\longrightarrow$	A
		$0, 1$	$\longleftarrow$	$0, 1$
		$R_0 \leftarrow Z_q^{m \times m}$		
R'_0, R'_1	$\longleftarrow$	$R'_0 = R_0, R'_1 = 0^{m \times m}$		
$b_1 \leftarrow \{0, 1\}$				
$U(R'_{b_1})$	$\longrightarrow$	$U(R'_{b_1})$	$\longrightarrow$	$U(R'_{b_1})$
		$C(0, R_0) = AR_0$	$\longrightarrow$	$C(0, R_0)$
				$A, C(0, R_0), U(R'_{b_1})$
b'_1	$\longleftarrow$	b'_1	$\longleftarrow$	Guess b_1 as b'_1
if $b_1 = b'_1$	you win $\longrightarrow$		you win $\longrightarrow$	
else	you lose $\longrightarrow$		you lose $\longrightarrow$	

MFHE Security Game 2: if $b_2 = 0$, we are in Hybrid 1. if $b_2 = 1$, we are in Hybrid 2

GSW Challenger		**Reduction**		**MFHE Adversary**
$pk = A$	$\longrightarrow$	A	$\longrightarrow$	A
$0, 1$	$\longleftarrow$	$0, 1$	$\longleftarrow$	$0, 1$
$b_2 \leftarrow \{0, 1\}$				
$R_0, R_1 \leftarrow \mathbb{Z}_q^{m \times m}$				
$C(b_2, R_{b_2}) = AR_{b_2} + b_2 G$	$\longrightarrow$	$C(b_2, R_{b_2})$	$\longrightarrow$	$C(b_2, R_{b_2})$
		$U(0)$	$\longrightarrow$	$U(0)$
				$A, C(b_2, R_{b_2}), U(0)$
b_2'	$\longleftarrow$	b_2'	$\longleftarrow$	Guess b_2 as b_2'
if $b_2 = b_2'$	you win $\longrightarrow$		you win $\longrightarrow$	
else	you lose $\longrightarrow$		you lose $\longrightarrow$	

MFHE Security Game 3: if $b_3 = 0$, we are in Hybrid 2. if $b_3 = 1$, we are in Hybrid 3

GSW Challenger		**Reduction**		**MFHE Adversary**
$pk = A$	$\longrightarrow$	A	$\longrightarrow$	A
		$0, 1$	$\longleftarrow$	$0, 1$
		$R_1 \leftarrow Z_q^{m \times m}$		
R_0', R_1'	$\longleftarrow$	$R_0' = 0^{m \times m}, R_1' = R_1$		
$b_3 \leftarrow \{0, 1\}$				
$U(R_{b_3}')$	$\longrightarrow$	$U(R_{b_3}')$	$\longrightarrow$	$U(R_{b_3}')$
		$C(1, R_1) = AR_1 + G$	$\longrightarrow$	$C(1, R_1)$
				$A, C(1, R_1), U(R_{b_3}''')$
b_3'	$\longleftarrow$	b_3'	$\longleftarrow$	Guess b_3 as b_3'
if $b_3 = b_3'$	you win $\longrightarrow$		you win $\longrightarrow$	
else	you lose $\longrightarrow$		you lose $\longrightarrow$	

References

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. Cryptology ePrint Archive, Report 2012/078, 2012. https://eprint.iacr.org/2012/078.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. Cryptology ePrint Archive, Report 2011/344, 2011. https://eprint.iacr.org/2011/344.

[Gen09a] Craig Gentry. A fully homomorphic encryption scheme, 2009. https://crypto.stanford.edu/craig/craig-thesis.pdf.

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Stoc*, volume 9, pages 169–178, 2009.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. Cryptology ePrint Archive, Report 2013/340, 2013. https://eprint.iacr.org/2013/340.

[MW15] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key fhe. Cryptology ePrint Archive, Report 2015/345, 2015. https://eprint.iacr.org/2015/345.