TFHE Overview

TFHE Deep Dive 系列的博客出自Ilaria Chillotti （即 TFHE 方案的第一作者）在 `zara` 上发布的一系列的技术博客， 并进行了报告，感兴趣的读者可以自行探索。 该篇博客是笔者学习 TFHE 算法总结性的 notes。

By the 1st author of TFHE , Ilaria Chillotti, TFHE Deep Dive Series posted in 2022:

- TFHE Deep Dive - Part I - Ciphertext types
- TFHE Deep Dive - Part II - Encodings and linear leveled operations
- TFHE Deep Dive - Part III - Key switching and leveled multiplications
- TFHE Deep Dive - Part IV - Programmable Bootstrapping

1 Notions Before TFHE

Remarks：

1. $\mathcal{R}=\mathbb{Z}[X]/\left(X^N+1\right)$ the ring of integer polynomials modulo the cyclotomic polynomial X^N+1, with N power of 2 . （定义在整数环上的多项式商环：模多项式为 X^N+1 ）
2. $\mathcal{R}_q=(\mathbb{Z}/q\mathbb{Z})[X]/\left(X^N+1\right)$, i.e., the same ring of integers $\mathcal{R}$ as above, but this time the coefficients are modulo q. Observe that we often note $\mathbb{Z}/q\mathbb{Z}$ as $\mathbb{Z}_q$. （定义在整数商环 $\mathbb{Z}/q\mathbb{Z}$ 上的多项式商环：模多项式为 X^N+1 ）
3. Balanced Mod : 整数商环 $\mathbb{Z}/q\mathbb{Z}$ 上的剩余系代表我们选取： $\{-\lfloor q/2\rfloor\ldots\lfloor q/2\rfloor\}$
4. 数字均用小写字母表示 $(a,b,m,s,\ldots)$, 多项式均用大写字母表示 $(A,B,M,S,\ldots)$.
5. 整数区间：$a\in\mathbb{Z}$ 到 $b\in\mathbb{Z}$ 记为 $[a..b]$.
6. MSB for Most Significant Bit and LSB for Least Significant Bit respectively.
7. 取整：$\lfloor\cdot\rceil$

2 PlainText And CipherText Space

- 明文模数： p ；明文空间 $M\in\mathcal{R}_p$
- 密文模数： q ；密文空间 $M\in\mathcal{R}_q$
- Scaling Factor ： $\Delta=\frac{q}{p}$

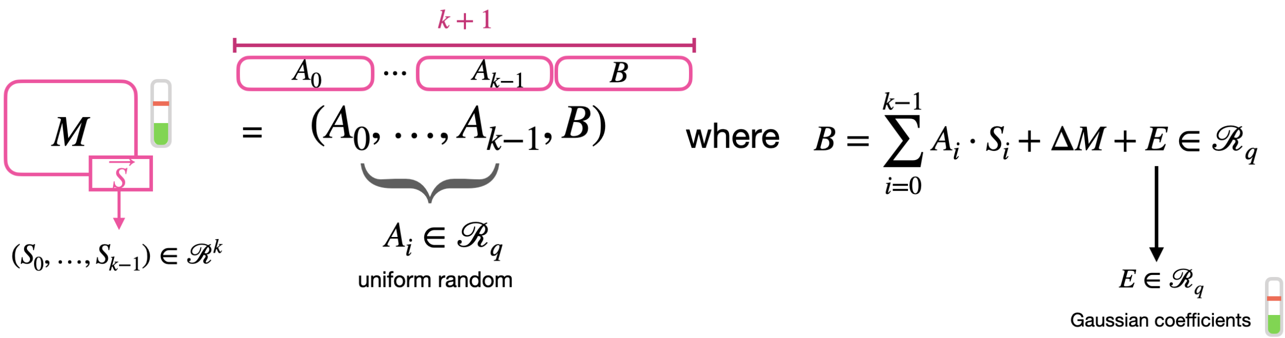
3 Ciphertext Types

$G \rightarrow General$: LWE + Ring-LWE

3.1 GLWE

密钥： $\vec{S} = (S_0, \dots, S_{k-1}) \in \mathcal{R}^k$ ； 明文： M

$$(A_0, \dots, A_{k-1}, B) \in GLWE_{\vec{S},\sigma}(\Delta M) \subseteq \mathcal{R}_q^{k+1} \tag{1}$$



Type : Vector of Polynomials (1-dimension)

记： $GLWE_{\vec{S},\sigma}(\Delta M)$ 为 GLWE 类型密文的一般表示 。

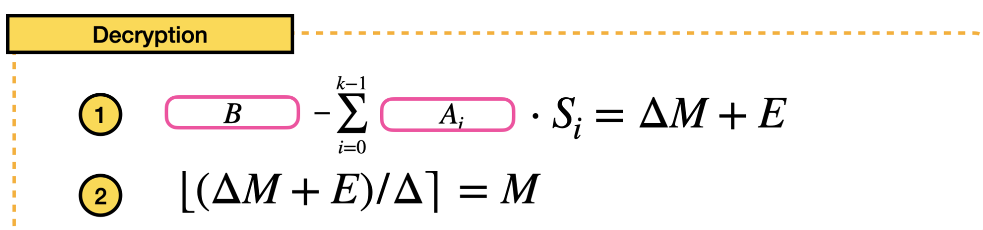


Figure : Decryption

3.2 GLev

密钥： $\vec{S} = (S_0, \dots, S_{k-1}) \in \mathcal{R}^k$ ； 明文： M

不同 scaling factor 下相同明文的 GLWE 密文向量:

$$\left(GLWE_{\vec{S},\sigma}\left(\frac{q}{\beta^1} M\right) \times \ldots \times GLWE_{\vec{S},\sigma}\left(\frac{q}{\beta^\ell} M\right) \right) = GLev_{\vec{S},\sigma}^{\beta,\ell}(M) \subseteq \mathcal{R}_q^{\ell\cdot(k+1)}. \tag{2}$$

Type : Vector of GLWE Ciphertexts (2-dimension)

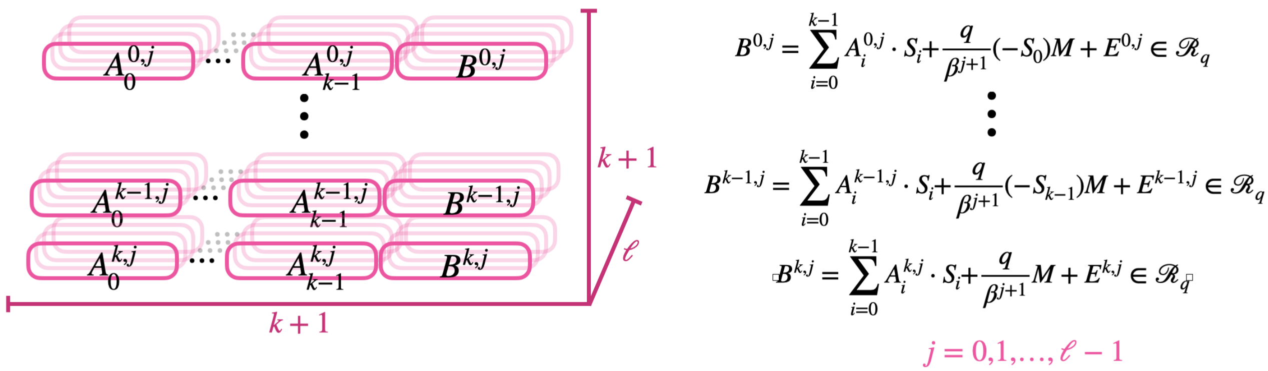
记： $GLev^{\beta,\ell}_{\vec{S},\sigma}(M)$ 为 GLev 类型密文的一般表示 。

3.3 GGSW

密钥： $\vec{S} = (S_0, \ldots, S_{k-1}) \in \mathcal{R}^k$ ； 明文： M

密钥每一维的 neg-polynomial（额外包含 1）与明文乘积的 GLev 密文

$$\left(GLev^{\beta,\ell}_{\vec{S},\sigma}(-S_0M) \times \ldots \times GLev^{\beta,\ell}_{\vec{S},\sigma}(-S_{k-1}M) \times GLev^{\beta,\ell}_{\vec{S},\sigma}(M)\right) = GGSW^{\beta,\ell}_{\vec{S},\sigma}(M) \subseteq \mathcal{R}_q^{(k+1)\times\ell(k+1)}. \tag{3}$$



Type : Vector of GLev Ciphertexts (3-dimension)

记： $GGSW^{\beta,\ell}_{\vec{S},\sigma}(M)$ 为 GGSW 类型密文的一般表示 。

4 Torus Visualization

Where is our message in TFHE $\mathcal{R}_p \to \mathcal{R}_q$? **Encoded in MSB !**

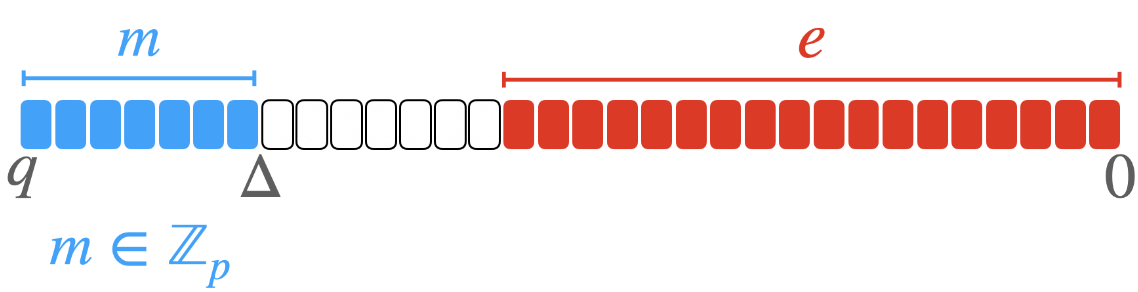


Figure : Message (mod p) lift to q

What if real numbers in a fixed interval ? **Encoded in MSB but mixed with errors !**

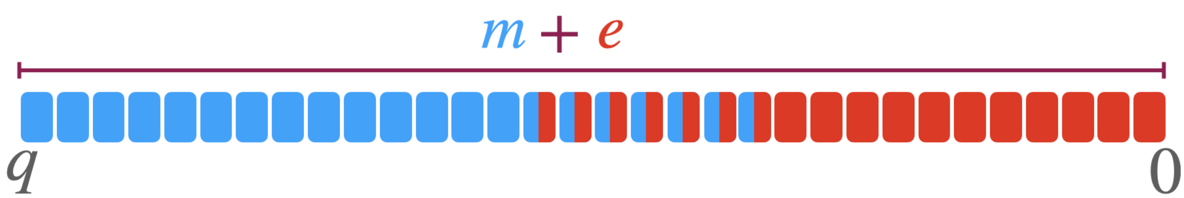


Figure : Message m: real numbers

环形结构 Torus, a mathematical structure that looks like a donut. Why this way ?

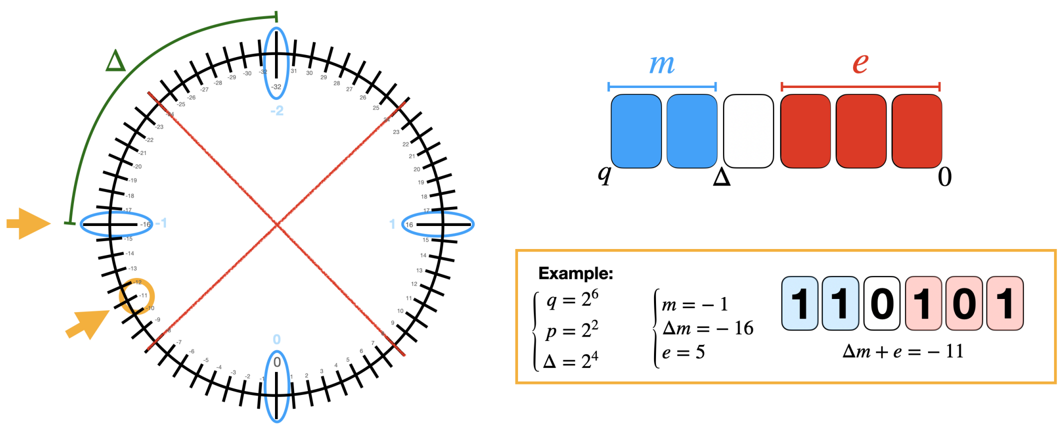


Figure : Torus Visualization for LWE

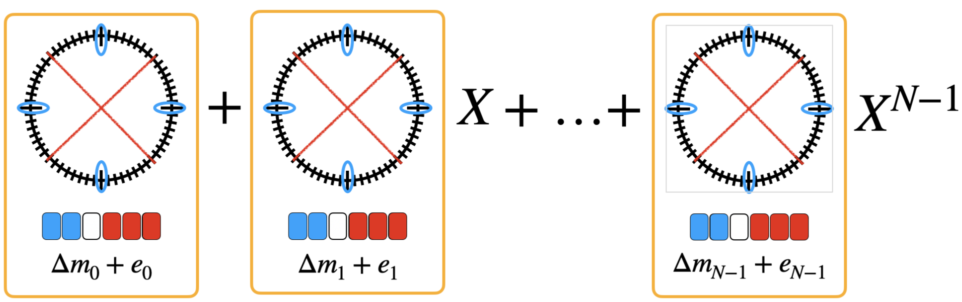


Figure : Torus Visualization for Ring-LWE

5 Building Blocks

Building Blocks $\rightarrow$ Fast Programmable Bootstrapping.

5.1 Homomorphic ADD

Message M, M' encrypted by the same key $\vec{S}$

$$\begin{aligned} C &= (A_0, \ldots, A_{k-1}, B) \in GLWE_{\vec{S},\sigma}(\Delta M) \subseteq \mathcal{R}_q^{k+1} \\ C' &= (A'_0, \ldots, A'_{k-1}, B') \in GLWE_{\vec{S},\sigma}(\Delta M') \subseteq \mathcal{R}_q^{k+1} \end{aligned} \tag{4}$$

Homomorphic ADD：

$$C^{(+)} = C + C' = (A_0 + A'_0, \ldots, A_{k-1} + A'_{k-1}, B + B') \in GLWE_{\vec{S},\sigma'}(\Delta(M + M')) \subseteq \mathcal{R}_q^{k+1} \tag{5}$$

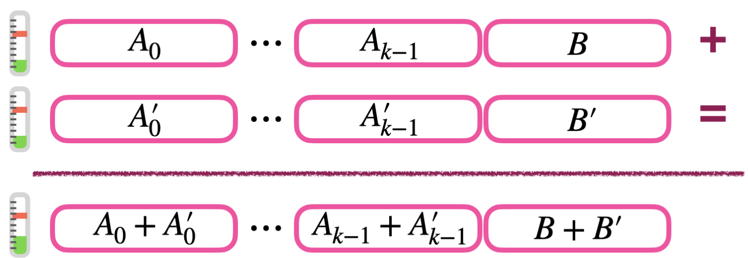


Figure : Homomorphic ADD

5.2 Homomorphic Mul Part 1

密钥： $\vec{S} = (S_0, \ldots, S_{k-1}) \in \mathcal{R}^k$ ；明文： M ； 密文： $C = (A_0, \ldots, A_{k-1}, B) \in GLWE_{\vec{S},\sigma}(\Delta M) \subseteq \mathcal{R}_q^{k+1}$.

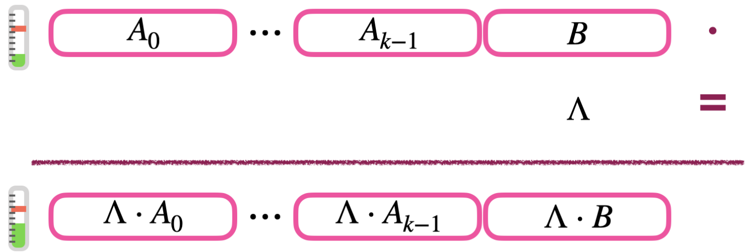
Little By Little , from constant-ciphertext multiplication to ciphertext-ciphertext multiplication !

- Homomorphic Mul by a **small constant polynomial**.

$$\Lambda = \sum_{i=0}^{N-1} \Lambda_i X^i \in \mathcal{R}. \tag{6}$$

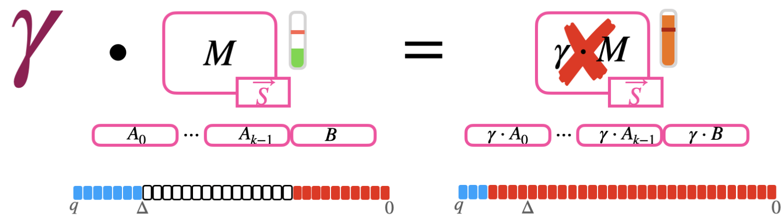
Mul：

$$C^{(\cdot)} = \Lambda \cdot C = (\Lambda \cdot A_0, \ldots, \Lambda \cdot A_{k-1}, \Lambda \cdot B) \in GLWE_{\vec{S},\sigma''}(\Delta(\Lambda \cdot M)) \subseteq \mathcal{R}_q^{k+1} \tag{7}$$



- Homomorphic Mul by a **large constant**.

  If we directly times them :



  We need Decompose, Recompose :

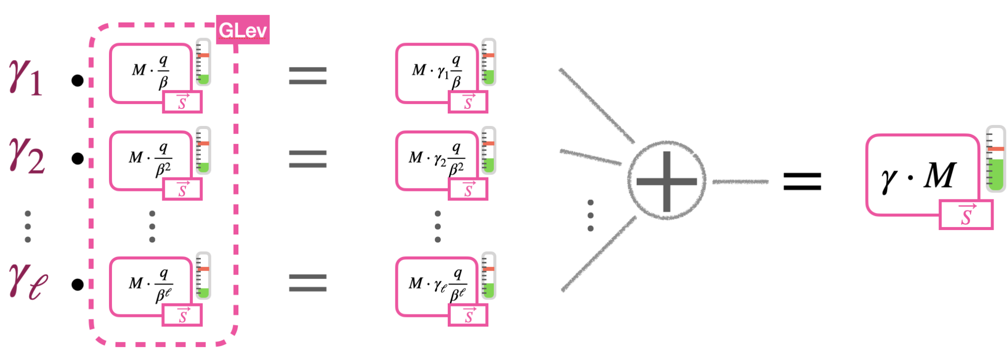
$$\gamma = \gamma_1 \frac{q}{\beta^1} + \gamma_2 \frac{q}{\beta^2} + \ldots + \gamma_\ell \frac{q}{\beta^\ell} \tag{8}$$

  We need GLev ciphertext :

$$\overline{C} = (C_1, \ldots, C_\ell) \in \left(GLWE_{\vec{S},\sigma}\left(\frac{q}{\beta^1} M\right) \times \ldots \times GLWE_{\vec{S},\sigma}\left(\frac{q}{\beta^\ell} M\right)\right) = GLev^{\beta,\ell}_{\vec{S},\sigma}(M) \subseteq \mathcal{R}_q^{\ell\cdot(k+1)}. \tag{9}$$

  Mul :

$$\langle \mathsf{Decomp}^{\beta,\ell}(\gamma), \overline{C}\rangle = \sum_{j=1}^{\ell} \gamma_j \cdot C_j \in GLWE_{\vec{S},\sigma'}(\gamma \cdot M) \subseteq \mathcal{R}_q^{k+1} \tag{10}$$



- Homomorphic Mul by a **large constant polynomial**.

  Almost the same with the constant one. Decompose the polynomial this time:

$$\mathsf{Decomp}^{\beta,\ell}(\Lambda) = (\Lambda^{(1)}, \ldots, \Lambda^{(\ell)}) \tag{11}$$

  where $\Lambda^{(j)} = \sum_{i=0}^{N-1} \Lambda_{i,j} \cdot X^i$, with $\Lambda_{i,j} \in \mathbb{Z}_\beta$, such that:

$$\Lambda = \Lambda^{(1)}\frac{q}{\beta^1} + \ldots + \Lambda^{(\ell)}\frac{q}{\beta^\ell}. \tag{12}$$

PS：Decomp **实际上是一个升维过程，Recomp 是降维过程。**

Homomorphic Mul by a **ciphertext** ? Before this , we come into a similar process called Key Switching.

5.3 Key Switching

更换加密密钥： $GLWE_{\vec{S},\sigma}(\Delta M) \longrightarrow GLWE_{\vec{S}',\sigma}(\Delta M)$

原始密钥： $\vec{S} = (S_0, \ldots, S_{k-1}) \in \mathcal{R}^k$ ； 明文： M ； 原始密文： $C = (A_0, \ldots, A_{k-1}, B) \in GLWE_{\vec{S},\sigma}(\Delta M) \subseteq \mathcal{R}_q^{k+1}$.

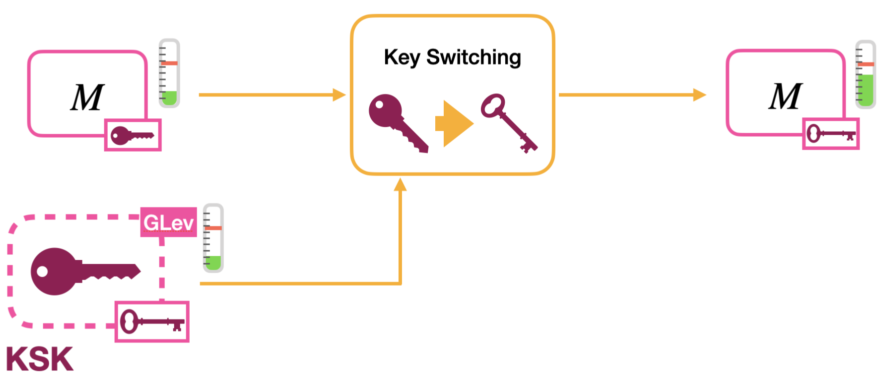
Key Switching Key

$$\mathsf{KSK}_i \in \left(GLWE_{\vec{S}',\sigma_{\mathsf{KSK}}}\left(\frac{q}{\beta^1}S_i\right) \times \ldots \times GLWE_{\vec{S}',\sigma_{\mathsf{KSK}}}\left(\frac{q}{\beta^\ell}S_i\right)\right) = GLev^{\beta,\ell}_{\vec{S}',\sigma_{\mathsf{KSK}}}(S_i) \subseteq \mathcal{R}_q^{\ell\cdot(k+1)} \tag{13}$$

思路：用 $\vec{S}'$ 加密后的 $\vec{S}$ 去执行解密的第一步，就得到了 $\vec{S}'$ 加密后的密文。实际上完整的 KSK 是一个 GGSW 密文。KeY Switching 与密文的同态乘法是很类似的。

Switching：

$$C' = \underbrace{\overbrace{(0, \ldots, 0, B)}^{\text{Trivial GLWE of } B} - \sum_{i=0}^{k-1} \overbrace{\langle \mathsf{Decomp}^{\beta,\ell}(A_i), \mathsf{KSK}_i \rangle}^{\text{GLWE encryption of } A_iS_i}}_{\text{GLWE encryption of } B - \sum_{i=0}^{k-1} A_iS_i = \Delta M + E} \in GLWE_{\vec{S}',\sigma'}(\Delta M) \subseteq \mathcal{R}_q^{k+1}. \tag{14}$$



Figure：Key Switching

5.4 Homomorphic Mul Part 2

Homomorphic Mul between two **ciphertexts** $\rightarrow$ called External Product in TFHE.

External Product

Setting : GLWE 密文 与 GGSW 密文

- a GLWE ciphertext encrypting a message $M_1 \in \mathcal{R}_p$ under the secret key $\vec{S} = (S_0, \ldots, S_{k-1}) \in \mathcal{R}^k$:

$$C = (A_0, \ldots, A_{k-1}, B) \in GLWE_{\vec{S},\sigma}(\Delta M_1) \subseteq \mathcal{R}_q^{k+1} \tag{15}$$

  where the elements A_i for $i \in [0..k-1]$ are sampled uniformly random from $\mathcal{R}_q$, and $B = \sum_{i=0}^{k-1} A_i \cdot S_i + \Delta M + E \in \mathcal{R}_q$, and $E \in \mathcal{R}_q$ has coefficients sampled from a Gaussian distribution χ_σ, as we have already seen before.

- a GGSW ciphertext encrypting a message $M_2 \in \mathcal{R}_p$ under the same secret key $\vec{S} = (S_0, \ldots, S_{k-1}) \in \mathcal{R}^k$:

$$\overline{\overline{C}} = (\overline{C}_0, \ldots, \overline{C}_{k-1}, \overline{C}_k) \in GGSW_{\vec{S},\sigma}^{\beta,\ell}(M_2) \subseteq \mathcal{R}_q^{(k+1)\times \ell(k+1)} \tag{16}$$

  where $\overline{C}_i \in GLev_{\vec{S},\sigma}^{\beta,\ell}(-S_i M_2)$ for $i \in [0..k-1]$ and $\overline{C}_k \in GLev_{\vec{S},\sigma}^{\beta,\ell}(M_2)$

Homomorphic Mul !

$$\begin{aligned} C' &= \overline{\overline{C}} \boxdot C = \langle \mathsf{Decomp}^{\beta,\ell}(C), \overline{\overline{C}} \rangle \\ &= \underbrace{\overbrace{\langle \mathsf{Decomp}^{\beta,\ell}(B), \overline{C}_k \rangle}^{\text{GLWE encrypt. of } BM_2} + \sum_{i=0}^{k-1} \overbrace{\langle \mathsf{Decomp}^{\beta,\ell}(A_i), \overline{C}_i \rangle}^{\text{GLWE encrypt. of } -A_i S_i M_2}}_{\text{GLWE encrypt. of } BM_2 - \sum_{i=0}^{k-1} A_i S_i M_2 \approx \Delta M_1 M_2} \in GLWE_{\vec{S},\sigma''}(\Delta M_1 M_2) \subseteq \mathcal{R}_q^{k+1} \end{aligned} \tag{17}$$

Figure : Homomophic Mul (External Product)

同时 Homomorphic Mul 和 Key Switching $\longrightarrow$ Functional Fey Switching

- GLWE 密钥不变，为 $\vec{S}$

- GGSW 的密文使用一个不同的密钥 $\vec{S}'$

得到： $GLWE_{\vec{S},\sigma''}(\Delta M_1 M_2)$

Internal Product

External Product in TFHE : $GLWE \times GGSW \rightarrow GLWE \rightarrow$ Internal Product : $GGSW \times GGSW \rightarrow GGSW$

Setting :

- a GGSW ciphertext encrypting a message $M_1 \in \mathcal{R}_p$ under the same secret key $\vec{S} = (S_0, \ldots, S_{k-1}) \in \mathcal{R}^k$:

$$\overline{\overline{C}}_1 = (\overline{C}_0, \ldots, \overline{C}_{k-1}, \overline{C}_k) \in GGSW^{\beta,\ell}_{\vec{S},\sigma}(M_1) \subseteq \mathcal{R}_q^{(k+1)\times\ell(k+1)}. \tag{18}$$

where, for $i \in [0..k-1]$:

$$\overline{C}_i = (C_{i,1}, \ldots, C_{i,\ell}) \in GLev^{\beta,\ell}_{\vec{S},\sigma}(-S_i M_1) \subseteq \mathcal{R}_q^{\ell\cdot(k+1)} \tag{19}$$

with $C_{i,j} \in GLWE_{\vec{S},\sigma}\left(\frac{q}{\beta^j}(-S_i M_1)\right)$ for $j \in [1..\ell]$ and

$$\overline{C}_k \in (C_{k,1}, \ldots, C_{k,\ell}) \in GLev^{\beta,\ell}_{\vec{S},\sigma}(M_1) \subseteq \mathcal{R}_q^{\ell\cdot(k+1)} \tag{20}$$

with $C_{k,j} \in GLWE_{\vec{S},\sigma}\left(\frac{q}{\beta^j}M_1\right)$ for $j \in [1..\ell]$

- a GGSW ciphertext encrypting a message $M_2 \in \mathcal{R}_p$ under the same secret key $\vec{S} = (S_0, \ldots, S_{k-1}) \in \mathcal{R}^k$:

$$\overline{\overline{C}} = (\overline{C}_0, \ldots, \overline{C}_{k-1}, \overline{C}_k) \in GGSW^{\beta,\ell}_{\vec{S},\sigma}(M_2) \subseteq \mathcal{R}_q^{(k+1)\times\ell(k+1)} \tag{21}$$

where $\overline{C}_i \in GLev^{\beta,\ell}_{\vec{S},\sigma}(-S_i M_2)$ for $i \in [0..k-1]$ and $\overline{C}_k \in GLev^{\beta,\ell}_{\vec{S},\sigma}(M_2)$

Homomorphic Mul !

$$\overline{\overline{C}}' = \overline{\overline{C}}_2 \boxtimes \overline{\overline{C}}_1 = (\overline{\overline{C}}_2 \boxdot C_{0,1}, \ldots, \overline{\overline{C}}_2 \boxdot C_{0,\ell}, \ldots, \overline{\overline{C}}_2 \boxdot C_{k,1}, \ldots, \overline{\overline{C}}_2 \boxdot C_{k,\ell}). \tag{22}$$

The result is :

$$\overline{\overline{C}}' = \overline{\overline{C}}_2 \boxtimes \overline{\overline{C}}_1 \in GGSW^{\beta,\ell}_{\vec{S},\sigma''}(M_1 M_2) \subseteq \mathcal{R}_q^{(k+1)\times\ell(k+1)}. \tag{23}$$

Figure : Homomophic Mul (Internal Product)

5.5 CMux

The CMux operation is the homomorphic version of a Mux gate, also known as multiplexer gate.

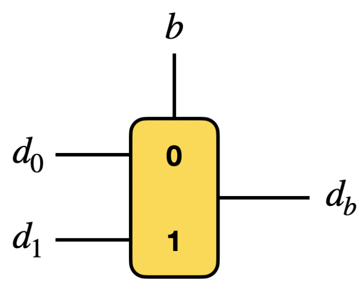


Figure : CMux Gate

简单来说一次乘法、两次加法即可实现：

$$b \cdot (d_1 - d_0) + d_0 = d_b \tag{24}$$

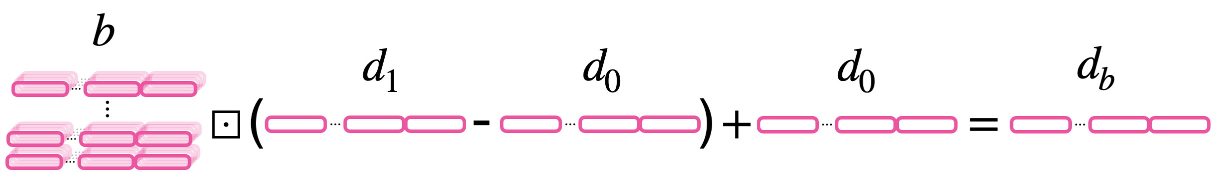


Figure : CMux Operation

5.6 Modulus Switching

密文空间变换： $\mathcal{R}_q \rightarrow \mathcal{R}_w$

Let p and q be two positive integers (powers of 2 for simplicity), such that $p \leq q$ and let $\Delta = q/p$. Let's recall that an LWE ciphertext encrypting a message $m \in \mathbb{Z}_p$ under the secret key $\vec{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{Z}^n$ is a tuple:

$$c = (a_0, \ldots, a_{n-1}, b) \in LWE_{\vec{s},\sigma}(\Delta m) \subseteq \mathbb{Z}_q^{n+1} \tag{25}$$

The Modulus Switching from q to w is easy :

$$\tilde{c} = (\tilde{a}_0, \ldots, \tilde{a}_{n-1}, \tilde{a}_n = \tilde{b}) \in LWE_{\vec{s},\sigma}(\tilde{\Delta} m) \subseteq \mathbb{Z}_\omega^{n+1},\ where\ \tilde{a}_i = \left\lfloor \frac{\omega \cdot a_i}{q} \right\rceil \in \mathbb{Z}_\omega. \tag{26}$$

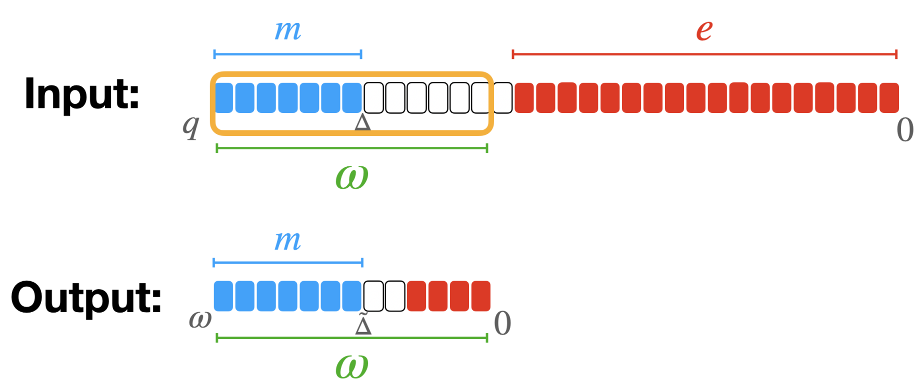
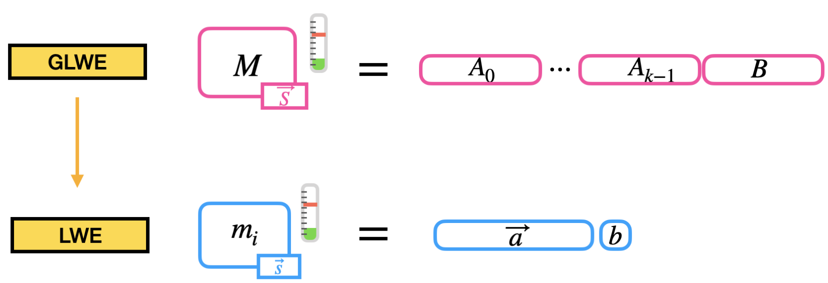


Figure : Modulus Switching

5.7 Sample Extraction

A sample extraction is an operation that takes as input a GLWE ciphertext, encrypting a polynomial message, and *extracts* the encryption of one of the coefficients of the message as a LWE ciphertext.

即 **提取 GLWE 加密的多项式其中的一个系数加密后的结果（LWE 密文）。**



Let's take a GLWE ciphertext encrypting a message $M = \sum_{j=0}^{N-1} m_j X^j \in \mathcal{R}_p$ under secret key:

$$\vec{S} = (S_0 = \sum_{j=0}^{N-1} s_{0,j} X^j, \ldots, S_{k-1} = \sum_{j=0}^{N-1} s_{k-1,j} X^j) \in \mathcal{R}^k \tag{27}$$

The ciphertext :

$$C = \left(A_0 = \sum_{j=0}^{N-1} a_{0,j} X^j, \ldots, A_{k-1} = \sum_{j=0}^{N-1} a_{k-1,j} X^j, B = \sum_{j=0}^{N-1} b_j X^j \right) \in GLWE_{\vec{S},\sigma}(\Delta M) \subseteq \mathcal{R}_q^{k+1} \tag{28}$$

从上述密文中提取 加密多项式 M 的第 h 个系数:

- 加密密钥： $\vec{s} = (s_{0,0}, \ldots, s_{0,N-1}, \ldots, s_{k-1,0}, \ldots, s_{k-1,N-1}) \in \mathbb{Z}^{kN}$.
- m_j 的 LWE 密文 $c = (a_0, \ldots, a_{n-1}, b) \in \mathbb{Z}_q^{n+1}$

$$\begin{cases} a_{N \cdot i+j} \leftarrow a_{i,h-j} & \text{for } 0 \le i < k, 0 \le j \le h \\ a_{N \cdot i+j} \leftarrow -a_{i,h-j+N} & \text{for } 0 \le i < k, h+1 \le j < N \\ b \leftarrow b_h \end{cases} \tag{29}$$

5.8 Blind Rotation

The core of the fast bootstrapping!

How to rotate :

- Rotate the coefficients of a polynomial ?

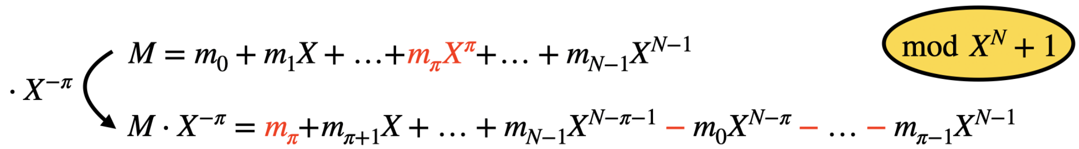
  Shift the coefficients towards the left (or the right) : $a_i \cdot x^i \to a_{i+h} \cdot x^i$

  How ?

$$M = m_0 + m_1 X + m_2 X^2 + \ldots + m_\pi X^\pi + \ldots + m_{N-1} X^{N-1} \in \mathcal{R}_q \tag{30}$$

  We multiply it times $X^{-\pi} \in \mathcal{R}_q$

$$M \cdot X^{-\pi} = m_\pi + m_{\pi+1} X + \ldots + m_{N-1} X^{N-\pi-1} - m_0 X^{N-\pi} - \ldots - m_{\pi-1} X^{N-1} \in \mathcal{R}_q. \tag{31}$$

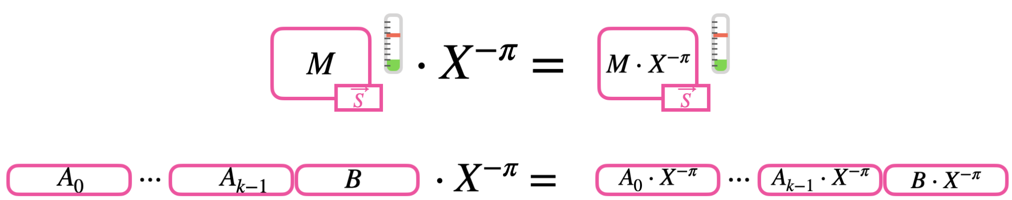


- Rotate the coefficients of an encrypted polynomial $\in GLWE$?

$$C = (A_0, \ldots, A_{k-1}, B) \in GLWE_{\vec{S},\sigma}(\Delta M) \subseteq \mathcal{R}_q^{k+1} \tag{32}$$

  Rotating :

$$C^{(\mathsf{rot})} = \left(A_0 \cdot X^{-\pi}, \ldots, A_{k-1} \cdot X^{-\pi}, B \cdot X^{-\pi}\right) \in GLWE_{\vec{S},\sigma}(\Delta M \cdot X^{-\pi}) \subseteq \mathcal{R}_q^{k+1}. \tag{33}$$



Blind Rotation ? $\to$ We wanna hide the shift π !

Binary decomposition :

$$\pi = \pi_0 + \pi_1 \cdot 2 + \pi_2 \cdot 2^2 + \ldots + \pi_\delta \cdot 2^\delta \quad where\ \delta = \log_2(N) \tag{34}$$

Then $M \cdot X^{-\pi}$:

$$\begin{aligned} M \cdot X^{-\pi} &= M \cdot X^{-\pi_0 - \pi_1 \cdot 2 - \pi_2 \cdot 2^2 + \ldots - \pi_\delta \cdot 2^\delta} \\ &= M \cdot X^{-\pi_0} \cdot X^{-\pi_1 \cdot 2} \cdot X^{-\pi_2 \cdot 2^2} \cdot \ldots \cdot X^{-\pi_\delta \cdot 2^\delta}. \end{aligned} \tag{35}$$

Compute $M \cdot X^{-\pi_j \cdot 2^j} \to$ CMux

$$M \cdot X^{-\pi_j \cdot 2^j} = \begin{cases} M & \text{if } \pi_j = 0 \\ M \cdot X^{-2^j} & \text{if } \pi_j = 1 \end{cases} \tag{36}$$

The full process of Blind Rotation

We need ciphertexts as follows :

- a GGSW encryption of π_j ;
- a GLWE encryption of M as the "0" option ;
- a GLWE encryption of $M \cdot X^{-2j}$ (rotation of a clear number of positions) as the "1" option.

One CMux :

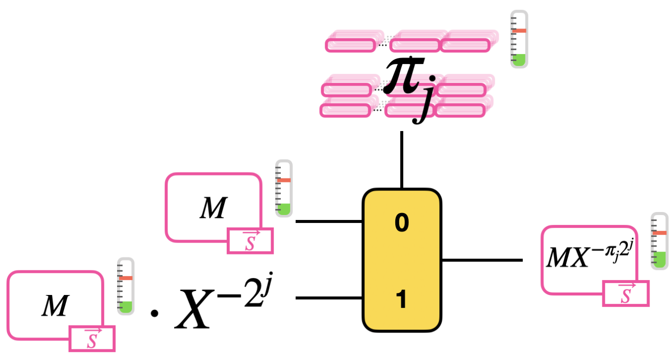


Figure : Single Rotation

The whole blind rotation :

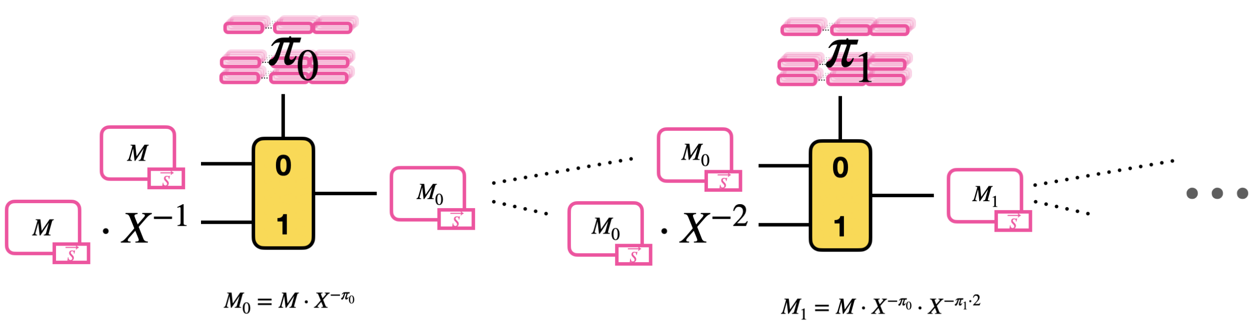


Figure : Blind Rotation

6 Bootstrapping

What is Bootstrapping ? **Evaluate the decryption function homomorphically**

在 LWE 类问题中，解密算法可以分为下面两步：

1. **STEP-1** : The computation of the linear combination :

$$b-\sum_{i=0}^{n-1} a_i s_i=\Delta m+e \in \mathbb{Z}_q \tag{37}$$

2. **STEP-2** : The rescale and rounding : $\left\lfloor \frac{\Delta m+e}{\Delta} \right\rceil = m$

其中 **STEP-1** 是简单的，用同态加/乘法足以解决，而真正做到减低噪声是第二步，这一步非常关键，也往往是 Bootstrapping 里面最耗费时间的操作。在 TFHE 中，最大的突破在于实现了 Fast Bootstrapping。

STEP-1 是简单的，因此这里首先讨论在 TFHE 中如如何进行 **STEP-2**： 将第一步得到的结果 $\pi=b-\sum_{i=0}^{n-1} a_i s_i=\Delta m+e$ 作为 X 的指数，即 $X^{-\pi}$ 去 Blind Rotate 一个 Look-Up Table (LUT：**evaluates the second step of the decryption (rescale and rounding).**)

$$LWE_{\vec{s},\sigma}(\Delta m) \times BK(\in GGSW) \rightarrow LWE_{\vec{s},\sigma}(\Delta m) \tag{38}$$

6.1 Step 2 Blind Rotation in Bootstrapping

Associate the value m with the plaintext $\Delta m+e$, the message distribution in q:

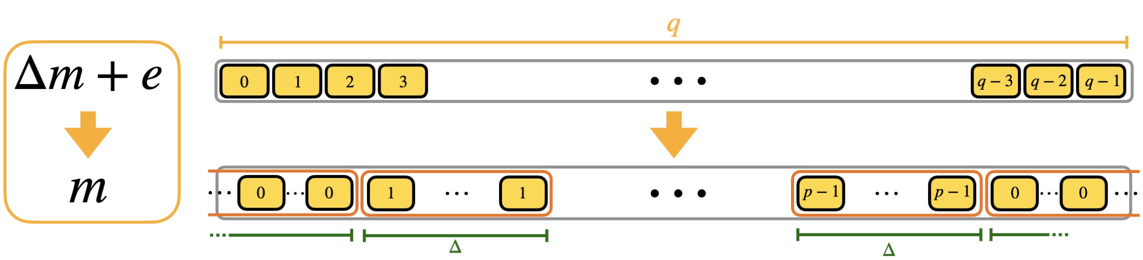


Figure : Mega-cases

We call the blocks containing multiple repetitions of the same value *mega-cases*.

The way we evaluate such LUT is by performing a polynomial rotation: the idea is to put all the elements of the redundant LUT into a polynomial and rotate the polynomial $\Delta m+e$ by multiplying $X^{-(\Delta m+e)}$. The rotation has the effect to bring one of the elements contained in the mega-case corresponding to m in the constant position of the polynomial.

将 LUT 表存入一个多项式，用 Blind Rotation 操作去 rotate (乘$X^{-(\Delta m+e)}$)，常数项上的系数就对应解密的 m ，即 reading position。

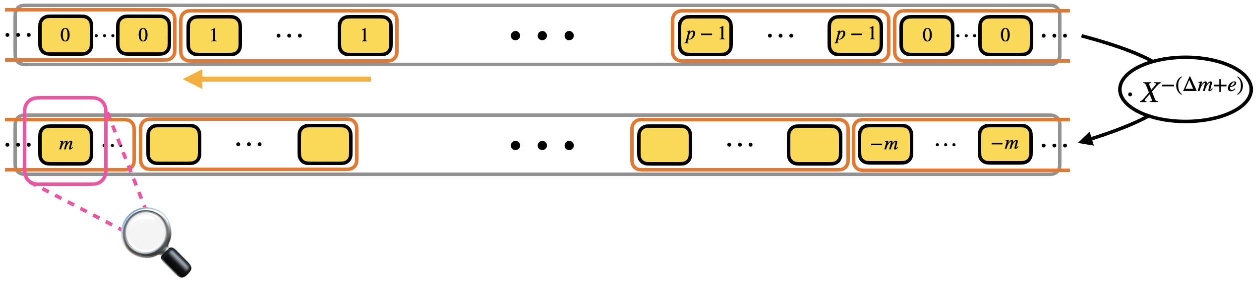


Figure : Reading Position

How to store such LUT in detail ?

- TFHE 里我们操作的多项式均模 X^N+1， 即 最多 N 个系数 （通常 $N<q: N=2^{10}, q=2^{32}$）。我们肯定需要进行数据压缩 modulus switching。
- 考虑 单项式 X 在多项式商环上的阶，为 ： $2*N$，$X^{2N}\equiv 1 \mod (X^N+1)$ 。也就是说，利用 rotation，我们最多得到 $2N$ 个结果 $(X^a = X^{a+2N} \mod (X^N+1))$。于是我们要把 $[0..q-1]$ 上的 $\Delta M+e$ 对应信息 转换到 $2\cdot N$ 上的元素内。（if **negacyclic property** exists，N is OK without padding bit）

Finally, **the modulus switching is going to be done from q to $2N$ ！**

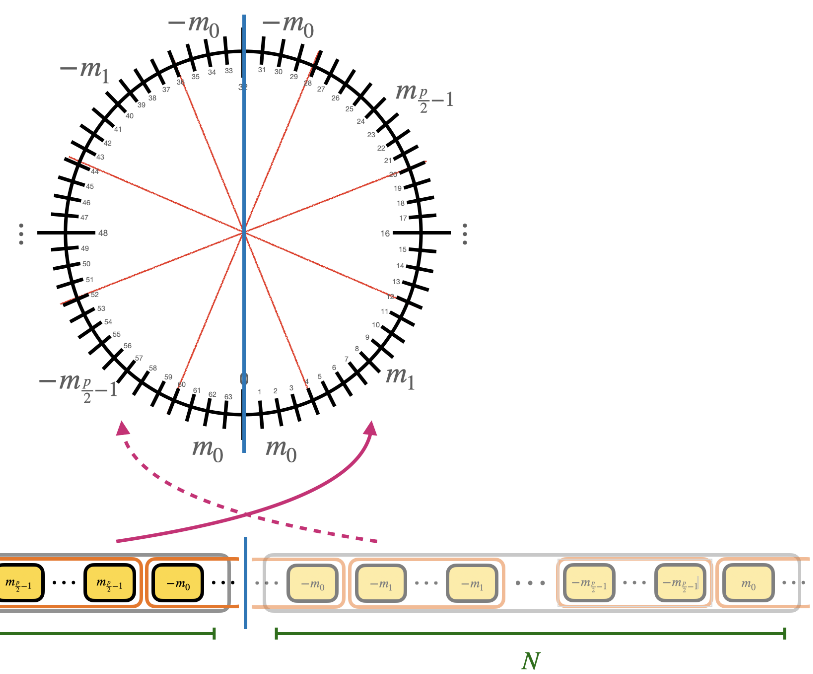


Figure : Modulus Switching for LUT

6.2 Step 1 + 2 Bootstrapping

Need :

- $c = (a_0, \ldots, a_{n-1}, b) \in LWE_{\vec{s},\sigma}(\Delta m) \subseteq \mathbb{Z}_q^{n+1}$
- LUT polynomial $V \in \mathcal{R}_q$
- **Bootstrapping key** : GGSW encryptions of $\vec{s} = (s_0, \ldots, s_{n-1})$ under a new GLWE secret key : S'

$$\mathsf{BK} = (\mathsf{BK}_0, \ldots, \mathsf{BK}_{n-1}) \quad \textit{where } \mathsf{BK}_i \in GGSW^{\beta,\ell}_{\vec{S}',\sigma}(s_i) \subseteq \mathcal{R}_q^{(k+1)\times\ell(k+1)} \tag{39}$$

Process

1. Modulus switching :

   Step 2 需要的 exponential information 是限定在 2N 范围内的，于是 $q \to 2N$ ：

$$c = (a_0, \ldots, a_{n-1}, b) \in LWE_{\vec{s},\sigma}(\Delta m) \subseteq \mathbb{Z}_q^{n+1} \longmapsto \tilde{c} = (\tilde{a}_0, \ldots, \tilde{a}_{n-1}, \tilde{b}) \in LWE_{\vec{s},\sigma'}(\tilde{\Delta} m) \subseteq \mathbb{Z}_{2N}^{n+1} \tag{40}$$

2. Blind rotation

   - Initialize the blind rotation by multiplying the trivial GLWE encryption $V \cdot X^{-\tilde{b}}$ (rotation)
   - Pass the trivial encryption of $V \cdot X^{-\tilde{b}} = V_0$ as input to a first CMux :
     - Selector : the GGSW encryption of the bit s_0
     - Options : V_0 and $V_0 \cdot X^{\tilde{a}_0}$
     - Output : a GLWE encryption of $V_1 = V_0 \cdot X^{\tilde{a}_0 s_0}$
   - Pass the GLWE encryption of V_1 to the second CMux :
     - Selector : the GGSW encryption of the bit s_1
     - Options : V_1 and $V_1 \cdot X^{\tilde{a}_1}$
     - Output : a GLWE encryption of $V_1 = V_0 \cdot X^{\tilde{a}_0 s_0}$

     Until N CMuxes done.
   - Final Output :

$$\begin{aligned} V_n &= V_{n-1} \cdot X^{\tilde{a}_{n-1} s_{n-1}} \\ &= \ldots \\ &= V \cdot X^{-\tilde{b}} \cdot X^{\tilde{a}_0 s_0} \cdot \ldots \cdot X^{\tilde{a}_{n-1} s_{n-1}} \\ &= V \cdot X^{-\tilde{b} + \sum_{i=0}^{n-1} \tilde{a}_i s_i} = V \cdot X^{-(\tilde{\Delta} m + \tilde{e})} \end{aligned} \tag{41}$$

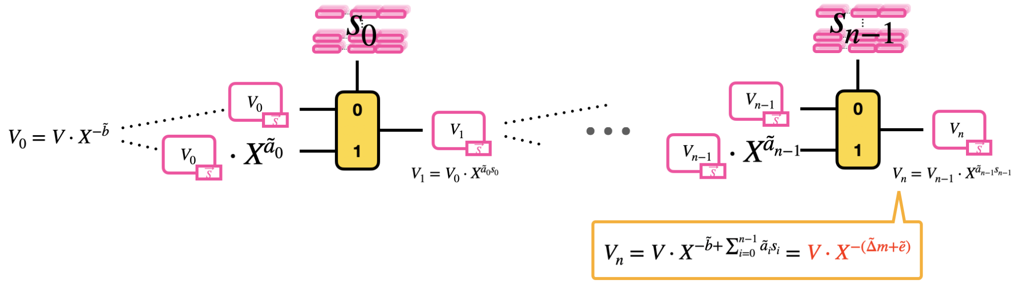


Figure : Blind Rotation

3. Sample Extraction : extract the constant sample, that is $LWE_{\vec{S}',\sigma}(\Delta m)$.

4. (Key switching : $LWE_{\vec{S}',\sigma}(\Delta m) \to LWE_{\vec{s},\sigma}(\Delta m)$)

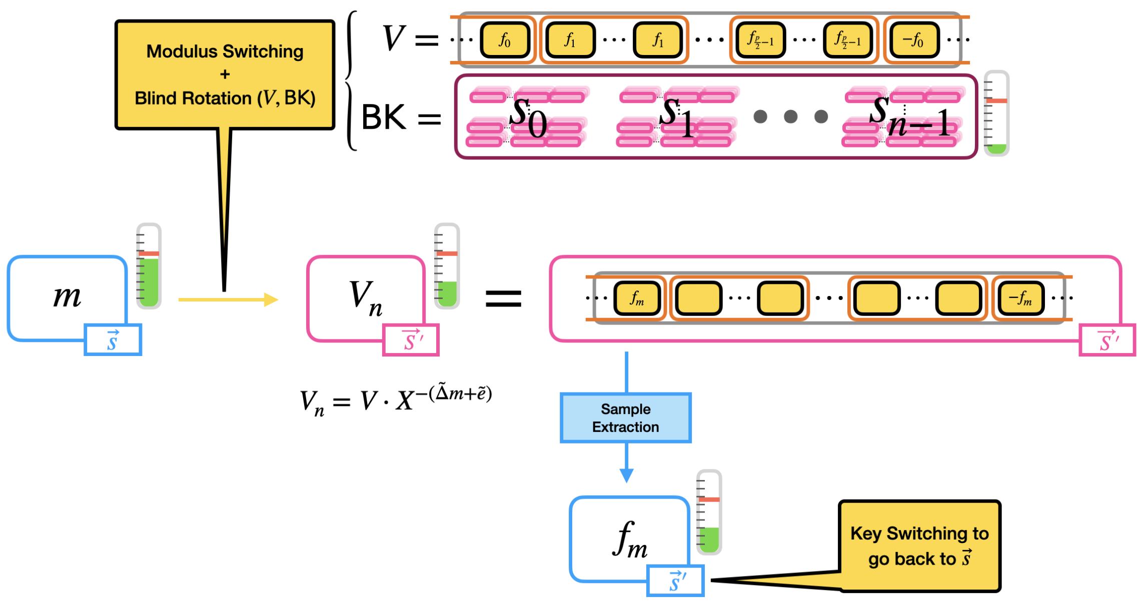


Figure : Bootstrapping

Why programmable ?

The V polynomial is actually a table. Do operations f on V and we obtain a new table V_f encode $f(m)$.

7 Non-arithmetic Operation

An example of bootstrapping: the Gate Bootstrapping ! Non-arithmetic Operation can be implemented by the programmable LUT.

所有的比特门运算 (AND, NAND, OR, XOR, etc.)，都可以通过 programmable bootstrapping 实现。这里以 ADD 门为例，构造 **ADD Gate Bootstrapping** 。

ADD Gate Bootstrapping Construction

- Input： two LWE ciphertexts c_1, c_2 encrypting two bits μ_1, μ_2 under the same secret key.
    - $c_1 = LWE_{s,\sigma}(\Delta\mu_1)$
    - $c_2 = LWE_{s,\sigma}(\Delta\mu_2)$
- Encoding info : $0 \to -q/8; 1 \to q/8$
- Process :
    1. 线性运算，取决于需要计算的函数。AND Gate 使用：

$$\mu_1 + \mu_2 - q/8 \tag{42}$$

2. 基于 LUT 进行Bootstrapping，LUT 对应多项式为 $V = \sum_{j=0}^{N-1} \frac{q}{8} X^j$

   V is rescaled sign function：对所有正数输入得到 q/8，对所有负数输入得到 - q/8

Cleartext		**Encoded**		**Linear combination of encodings**	**Bootstrapping**	**Expected encoded result**	**Expected cleartext result**
μ_1	μ_2	μ_1	μ_2	$\mu_1 + \mu_2 - q/8$		$\mu_1 \wedge \mu_2$	$\mu_1 \wedge \mu_2$
0	0	$-q/8$	$-q/8$	$-3q/8$		$-q/8$	0
0	1	$-q/8$	$q/8$	$-q/8$	$\xrightarrow{-q/8}$	$-q/8$	0
1	0	$q/8$	$-q/8$	$-q/8$		$-q/8$	0
1	1	$q/8$	$q/8$	$q/8$	$\xrightarrow{q/8}$	$q/8$	1

Figure : ADD Gate Bootstrapping

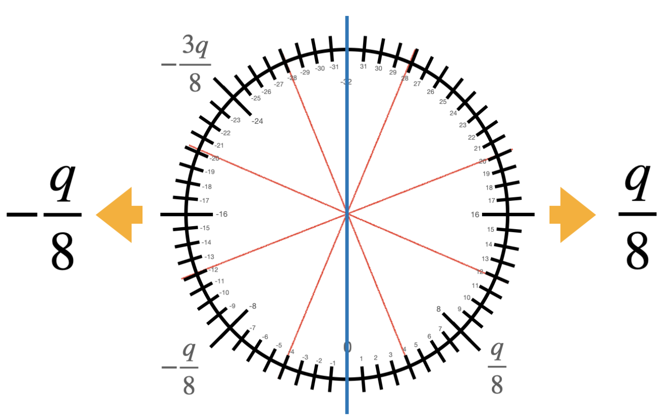


Figure : Torus View

由于 TFHE 很好地支持了非算术运算，因此 TFHE 中 Gate Bootstrapping 中可以用于神经网络的非线性激活函数构造。