

本文主要记录隐私计算中的同态加密（Homomorphic Encryption, HE）技术，包括部分同态加密（RSA、GM、ElGamal、Paillier）、近似同态加密（BGN）、有限级数全同态加密和全同态加密（DGHV、BGV、BFV、CKKS、GSW、FHEW、TFHE）等技术，**仅供参考**。

由于篇幅限制，将同态加密技术的介绍分为四个部分，**第一部分**讲述部分同态加密和近似同态加密技术；**第二部分**讲述 Bootstrapping 和 DGHV、BGV 全同态加密方案；**第三部分**讲述 SIMD 打包技术与 BFV、CKKS 全同态加密方案；**第四部分**讲述 GSW、FHEW、TFHE 全同态加密方案。

在本文中为**简化表示**，将加密记为 $Enc(\cdot)$ ，解密记为 $Dec(\cdot)$ ，不标明使用的公钥和私钥代表只有一对公私钥。本文中介绍的加密方案基本都依赖于各种**计算难题**，在之前的文章中有过介绍。

四、（有限级数）全同态加密（续）

7. GSW 加密方案

GSW 方案由 Gentry、Sahai 和 Waters 发表^[14]，开创了第三代全同态加密方案。GSW 加密方案创新性地提出了和第二代全同态加密方案**不同的加密路径**，主要思想是利用**近似矩阵特征值**的特性，以更复杂的密文为代价，实现较小噪声增长的乘法同态运算。由于噪声的增长与明文空间大小有关，GSW 方案及之后的优化方案多集中于**逻辑电路**的全同态运算（即明文空间为 $m \in \{0, 1\}$ ）。

论文中呈现的 GSW 方案**并非在多项式环上**进行加密，本文首先介绍原论文方案，随后介绍 GSW 方案在多项式环上的衍生方案。

首先给出论文中定义的三种运算，对于向量 $a = (a_1, \dots, a_n) \in \mathbb{Z}_q^n$ ，设 $\ell = \lfloor \log_2 q \rfloor + 1$ 代表元素二进制长度（更通用的可以设为 B 进制，这里方便表示统一**以二进制为例**），有如下三种运算：

1. BitDecomp：该运算将向量所有元素进行**二进制分解**：

$$BitDecomp(a) = a' = (a_{1,0}, \dots, a_{1,\ell-1}, a_{2,0}, \dots, a_{2,\ell-1}, \dots, a_{n,\ell-1})$$

其中 $a_i = \sum_{j=0}^{\ell-1} (a_{i,j} \cdot 2^j)$ ，其**逆运算**为重新组合： $BitDecomp^{-1}(a') = a$ 。

2. Powersof2：该运算将向量所有元素**填充2次幂**：

$$Powersof2(a) = (a_1 2^0, \dots, a_1 2^{\ell-1}, a_2 2^0, \dots, a_2 2^{\ell-1}, \dots, a_n 2^{\ell-1})$$

3. Flatten：该运算将向量**所有元素“铺平”**，即将元素大小分配均匀：

$$Flatten(a) = BitDecomp(BitDecomp^{-1}(a))$$

上述运算若应用到矩阵上则是对矩阵每一行分别进行运算。由上述运算的规则，对于 $a, b \in \mathbb{Z}_q^n$ 显然可以得到**向量内积**关系：

$$BitDecomp(a) \cdot Powersof2(b) = a \cdot b$$

对于 $a' \in \mathbb{Z}_q^{n\ell}, b \in \mathbb{Z}_q^n$ ，有：

$$a' \cdot Powersof2(b) = Flatten(a') \cdot Powersof2(b) = BitDecomp^{-1}(a') \cdot b$$

即 $Flatten(\cdot)$ **不改变**上述二进制组合下的向量内积结果。

7.1 密钥生成算法

1. 随机选取模数 q ，随机取 $s', e \in \mathbb{Z}_q^n$ ， $A' \in \mathbb{Z}_q^{m \times n}$ ，计算 $b = A's' + e \in \mathbb{Z}_q^n$ 。可以看出这是生成了 m 对 **LWE 问题实例**，以 s' 为密钥， e 为小噪声。（这里的表述为了与前文一致与论文不完全相同，本质是一样的。）
2. 公钥为 $A = (b, A') \in \mathbb{Z}_q^{m \times (n+1)}$ 为一个**横向拼接**在一起的矩阵，私钥为 $s = (1, -s') \in \mathbb{Z}_q^{(n+1)}$ ，显然有 $As = e$ 成立。

7.2 加密算法

设 $N = (n + 1)\ell$ ，且 $I_N \in \{0, 1\}^{N \times N}$ 为**单位矩阵**。对于明文 $\mu \in \mathbb{Z}_q$ ，随机选取 $R \in \{0, 1\}^{N \times m}$ ，使用公钥加密过程如下：

$$C = Enc(\mu) = Flatten(\mu I_N + BitDecomp(R \times A)) \in \mathbb{Z}_q^{N \times N}$$

其中 R 的存在只是为了**使密文随机**，不难验证 $R \times A$ 仍然是若干 **LWE 问题实例**组成的矩阵，满足 $R \times A \times s = R \times e$ 。 $Flatten$ 操作只是为了**使取值平均**，减少后续运算产生的噪声，在理解加密算法时可以忽视。

7.3 解密算法

使用私钥解密，首先计算 $C \times Powersof2(s)$ ，然后取其第 i 行的值 x_i （满足 $2^i \in (\frac{q}{4}, \frac{q}{2}]$ ），计算得到明文 $\mu = \lfloor \frac{x_i}{2^i} \rfloor$ 。

正确性

设 $v = Powersof2(s)$ ，可以观察到在解密时：

$$C \times v = \mu \cdot v + R \times A \times s = \mu \cdot v + R \times e = \mu v + e'$$

取第 i 行后得到 $\mu \cdot v_i + e'_i$ ，显然噪声 $\|e'_i\| = \|R_i \cdot e\| \leq \|e\|_1$ （范数 $\|\cdot\|$ 不标下标默认为**无穷范数**），当噪声大小不超过 $\frac{q}{8}$ 时，由于 $v_i \in (\frac{q}{4}, \frac{q}{2}]$ ，则 $\|\frac{e'_i}{v_i}\| < \frac{1}{2}$ ，因此解密可以成功。**噪声上限**为 $\frac{q}{8}$ 。

安全性

GSW 方案实际上使用了**若干组 LWE 问题**，实现了明文作为**近似特征值**的效果，即 $Cv = \mu v + e'$ （正常特征值满足 $Cv = \mu v$ ）。该组合本质上还是 LWE 问题，由于解决 LWE 问题是困难的，GSW 方案公开 A 作为公钥不会泄露 s 的取值，且在加密时 R 是随机选取的，可以认为 GSW 方案是安全的。

加法同态性

GSW 加密方案具有**加法同态性**，容易验证：

$$(C_0 + C_1)v = (\mu_0 + \mu_1)v + e'_0 + e'_1$$

且噪声增长较小。

乘法同态性

GSW 加密方案具有**乘法同态性**，容易验证：

$$(C_0 \times C_1)v = C_0 \times (\mu_1 v + Re_1) = \mu_0 \mu_1 v + \mu_1 e'_0 + C_0 e'_1$$

噪声变为 $e' = \mu_1 e'_0 + C_0 e'_1$ 。由于 C_0 进行过二进制分解，因此可以得到 $\|C_0 e'_1\| \leq N e_1$ 。所以噪声主要受 μ_1 即**明文空间大小**影响，若 GSW 只针对**单个比特**加密，则该方案同态乘法运算的噪声最多只增加 $N + 1$ 倍，相较于第二代同态加密大大减少，也**省去了密钥切换和模数切换操作**，但密文大小增加了大约 N 倍。

与非（NAND）同态性

在布尔电路中，考虑到**与非运算**（NAND）可以生成其他所有电路门，GSW 加密方案给出了与非同态计算方案实现布尔电路上的同态计算。考虑到在模 2 计算中，与非运算等价于 $1 - \mu_0 \mu_1$ ，因此与非同态计算可以验证如下：

$$(I_N - C_0 \times C_1)v = (1 - \mu_0 \mu_1)v - \mu_1 e'_0 - C_0 e'_1$$

本质上是**乘法同态**运算，噪声增长与乘法同态运算一致。

由于实现了**与非**同态计算就可以计算**任意电路门**，且经过每一个电路门噪声最多放大 $N + 1$ 倍，在不进行 Bootstrapping 运算的情况下，若初始噪声为 B ，则经过 L 个电路门后噪声上界变为 $B(N + 1)^L$ ，可以根据噪声上界衡量可计算**电路的深度**。自 GSW 方案后，很多论文关注于

快速的实现布尔电路计算（NAND 门）中的 **Bootstrapping 运算**，以达到全同态运算的目的，在后续 FHEW 和 TFHE 方案中会进行介绍。GSW 方案更多的是提供了一种**全新思路**，现今一般不直接使用 GSW 方案加密并计算（GSW 方案在数值计算上**密文较大**，且不支持第二代全同态加密方案中的 SIMD 批处理技术；在布尔计算上 FHEW 提出了一种更高效的 NAND 运算门），而更多的是作为布尔电路全同态计算中 **Bootstrapping 技术实现的关键部分**。

8. RGSW 加密方案

记 RGSW 为 GSW 方案的环变式，运算操作在**多项式环**之上，原理与 GSW 一致，主要应用在 Bootstrapping 技术的实现上，其原理介绍如下。

RGSW 方案本质上是若干 RLWE 方案的组合。在 RLWE 方案中，加密格式一般为（这里用 **BFV方案**中的加密方式）：

$$RLWE_s^{t/q}(m) = (a, as + e + \frac{q}{t}m)$$

其中 $a, s, e, m \in R_q$ 均为多项式环中的元素（ $R_q = \mathbb{Z}_q[x]/(x^N + 1)$ 为多项式环）， t 为明文空间中多项式系数模数， q 为密文空间模数（参考 BFV 方案），为了方便，设 $t|q$ ，因此 $\frac{q}{t}$ 可视为整数。

8.1 密钥生成算法

1. 设 q 为模数， B_g 为**分解基底**（在 GSW 方案中以二进制分解为例介绍，即 $B_g = 2$ ，此处也可以代入二进制理解）， $d_g = \lfloor \log_{B_g} q \rfloor + 1$ 为元素长度。生成重组矩阵 $G \in \mathbb{Z}_q^{2d_g \times 2}$ ：

$$G = \begin{bmatrix} B_g^0 & 0 \\ B_g^1 & 0 \\ \vdots & \vdots \\ B_g^{d_g-1} & 0 \\ 0 & B_g^0 \\ \vdots & \vdots \\ 0 & B_g^{d_g-1} \end{bmatrix}$$

2. 随机选取密钥 $s \in R_q$ ，并生成由 $2d_g$ 对 $RLWE_s(0)$ 密文组成的矩阵 $A \in R_q^{2d_g \times 2}$ ：

$$A = \begin{bmatrix} a_1 & a_1 s + e_1 \\ a_2 & a_2 s + e_2 \\ \vdots & \vdots \\ a_{2d_g} & a_{2d_g} s + e_{2d_g} \end{bmatrix}$$

3. 公钥为 A ，私钥为 s 。

8.2 加密算法

对于明文 $\mu \in \mathbb{Z}_q$ ，将其**明文编码**为 $m = x^\mu \bmod (x^N + 1)$ ，则显然 $m \in R_q$ ，其系数仅有一处为 1，其余为 0（这里的编码为**后续 FHEW 方案作铺垫**，并非一定要如此编码）。设缩放系数为 $u = \frac{q}{2}$ （假设 $2|q$ ，该系数可根据噪声要求调整），加密算法如下（为了辅助理解将缩放因子 u 也放入 $RLWE()$ 中，事实上该缩放因子是隐含在 $RLWE$ 方案中的）：

$$C = RGSW(\mu) = A + muG = \begin{bmatrix} a_1 + uB_g^0 m & a_1 s + e_1 \\ a_2 + uB_g^1 m & a_2 s + e_2 \\ \vdots & \vdots \\ a_{2d_g} & a_{2d_g} s + e_{2d_g} + uB_g^{d_g-1} m \end{bmatrix} = \begin{bmatrix} RLWE(-uB_g^0 m s) \\ \vdots \\ RLWE(-uB_g^{d_g-1} m s) \\ RLWE(uB_g^0 m) \\ \vdots \\ RLWE(uB_g^{d_g-1} m) \end{bmatrix}$$

其中有 $(a_i + uB_g^{i-1} m, a_i s + e_i) = RLWE(-uB_g^{i-1} m s)$ ，因为在解密时：

$$(a_i s + e_i) - (a_i + uB_g^{i-1} m)s = -uB_g^{i-1} m s + e_i$$

8.3 解密算法

使用私钥解密，取密文第 $d_g + 1$ 行 $RLWE(um)$ ，解密如下：

$$a_{d_g+1}s + e_{d_g+1} + um - a_{d_g+1}s = um + e_{d_g+1}$$

当 $\|\frac{e_{d_g+1}}{u}\| < \frac{1}{2}$ 时，对上述结果除去 u 并对系数**四舍五入取整**得到 m ，进而得到 μ （参考 BFV 解密正确性）。

正确性

正确性本质与 BFV 方案解密正确性一致。

安全性

RGSW 方案安全性依赖于 RLWE 问题，只要 RLWE 问题难解，RGSW 方案就可以认为是安全的。

加法同态性

RGSW 方案具有加法同态性，正确性容易验证，与 BFV 方案的加法同态性一致，且噪声增加较小，此处略过。

乘法同态性

RGSW 方案具有乘法同态性，这里借用 TFHE 方案中的表示方式，将 RGSW 方案的乘法分为**外积**和**内积**。

外积

RGSW 方案的外积可表示为 \boxtimes ： $RLWE \times RGSW \longrightarrow RLWE$ ，即**一个 RLWE 密文与一个 RGSW 密文运算得到一个 RLWE 密文**。

设 RLWE 密文为 $c = (a, b)$ ，对其进行分解后得到 $BitDecomp(c) = (a_0, \dots, a_{d_g-1}, b_0, \dots, b_{d_g-1})$ ，则同态外积计算如下（这里将缩放因子隐含在 $RLWE$ 方案内部）：

$$\begin{aligned} RLWE(m') \times RGSW(m) &= BitDecomp(c) \times C \\ &= (a_0, \dots, a_{d_g-1}, b_0, \dots, b_{d_g-1}) \times \begin{bmatrix} RLWE(-B_g^0 ms) \\ \vdots \\ RLWE(-B_g^{d_g-1} ms) \\ RLWE(B_g^0 m) \\ \vdots \\ RLWE(B_g^{d_g-1} m) \end{bmatrix} \\ &= \sum_{i=0}^{d_g-1} a_i \cdot RLWE(-B_g^i ms) + \sum_{i=0}^{d_g-1} b_i \cdot RLWE(B_g^i m) \\ &= \sum_{i=0}^{d_g-1} RLWE(-ms \cdot (a_i B_g^i) + m \cdot (b_i B_g^i)) \\ &= RLWE(m(b - as)) \\ &= RLWE(mm') \end{aligned}$$

上述推导过程中，为了便于理解，**省去了噪声的表示**，不难推导出噪声增加的部分主要是 $a_i e$ 和 $b_i e$ ，由于进行了进制分解，这样的噪声增加是可接受的。

内积

RGSW 方案的内积可表示为 \boxtimes ： $RGSW \times RGSW \longrightarrow RGSW$ ，即**两个 RGSW 密文运算得到一个 RGSW 密文**。

RGSW 的内积较为复杂，本质上由若干外积操作组成，计算如下（这里将缩放因子隐含在 $RLWE$ 方案内部）：

$$\begin{aligned}
RGSW(m_0) \times RGSW(m_1) &= \begin{bmatrix} RLWE(-B_g^0 m_0 s) \\ \vdots \\ RLWE(-B_g^{d_g-1} m_0 s) \\ RLWE(B_g^0 m_0) \\ \vdots \\ RLWE(B_g^{d_g-1} m_0) \end{bmatrix} \times RGSW(m_1) \\
&= \begin{bmatrix} RLWE(-B_g^0 m_0 s) \times RGSW(m_1) \\ \vdots \\ RLWE(-B_g^{d_g-1} m_0 s) \times RGSW(m_1) \\ RLWE(B_g^0 m_0) \times RGSW(m_1) \\ \vdots \\ RLWE(B_g^{d_g-1} m_0) \times RGSW(m_1) \end{bmatrix} \\
&= \begin{bmatrix} RLWE(-B_g^0 m_0 m_1 s) \\ \vdots \\ RLWE(-B_g^{d_g-1} m_0 m_1 s) \\ RLWE(B_g^0 m_0 m_1) \\ \vdots \\ RLWE(B_g^{d_g-1} m_0 m_1) \end{bmatrix} \\
&= RGSW(m_0 m_1)
\end{aligned}$$

上述推导过程中也省略了噪声的表示，可以看出最终密文每一行噪声的增长与 RGSW 外积的噪声增长一致。

9. FHEW 加密方案

FHEW 方案由 Ducas 和 Micciancio 发表^[15]，其主要创新在于提出一种**新型的** XAND 逻辑电路门的同态运算，并利用 RGSW 实现一个**累加器**，从而给出了该电路门 Bootstrapping 的构造方式，属于 Gate-Bootstrapping。

FHEW 方案利用 **LWE 问题**在整数环上加密，记 $LWE_s^{t/q}(m, E)$ 代表使用密钥向量 $s \in \mathbb{Z}_q^n$ 对明文 $m \in \mathbb{Z}_t$ 加密，加密噪声大小不超过 E 。为了便于表示，在本方案介绍中认为 $t|q$ ，即 $\frac{q}{t}$ 为**整数**。

9.1 密钥生成算法

FHEW 方案使用的加密方式为**对称加密**，因此只有私钥。设 q 为模数，随机取私钥 $s \in \mathbb{Z}_q^n$ 。

9.2 加密算法

对于明文 $m \in \{0, 1\}$ ，设明文空间大小 $t = 4$ ，且噪声限为 $E = \frac{q}{16}$ （和正常的 LWE 加密不同，为**与非同态计算**考虑）。随机取小噪声 e ，以及 $a \in \mathbb{Z}_q^n$ ，利用私钥加密得到：

$$c = Enc(m) = (a, b) = (a, as + e + \frac{q}{t}m)$$

9.3 解密算法

使用私钥解密流程如下：

$$m = Dec(c) = \lfloor \frac{t}{q}(b - as) \rfloor \bmod t$$

正确性

在解密过程中，有：

$$\frac{t}{q}(b - as) = \frac{t}{q}(e + \frac{q}{t}m) = \frac{t}{q}e + m$$

显然当噪声满足 $|\frac{t}{q}e| < \frac{1}{2}$ 时，**四舍五入会将其消掉**。由于加密要求 $|e| < \frac{q}{16}$ ，该条件显然成立，因此正确性得证。

安全性

FHEW 方案安全性依赖于 LWE 问题，若 LWE 问题难解，可以认为 FHEW 方案是安全的。

与非（XAND）同态性

FHEW 方案关注于布尔电路的同态计算，提出了一个新型 XAND 同态运算方式，该方式只需要用到**同态加法**，相较于 GSW 中使用同态乘法而言效率和空间上都得到较大优化。

新型 XAND 运算

注意到与非运算仅在 $m_0 = m_1 = 1$ 时结果为 0，其余结果均为 1，因此 FHEW 将其转换为：

$$\overline{m_0 m_1} = \lfloor \frac{5}{4} - \frac{1}{2}(m_0 + m_1) \rfloor$$

上述转换正确性容易验证，下面具体介绍 FHEW 方案中的与非同态计算。

该同态计算将 $LWE_s^{4/q}(m_0, \frac{q}{16})$ 和 $LWE_s^{4/q}(m_1, \frac{q}{16})$ 转换为 $LWE_s^{2/q}(\overline{m_0 m_1}, \frac{q}{4})$ ，计算方式如下：

$$c = LWE_s^{2/q}(\overline{m_0 m_1}, \frac{q}{4}) = (a, b) = (-a_0 - a_1, \frac{5q}{8} - b_0 - b_1)$$

正确性容易验证：

$$\begin{aligned} b - as &= \frac{5q}{8} - (b_0 + b_1 - a_0s - a_1s) \\ &= \frac{5q}{8} - \frac{q}{4}(m_0 + m_1) - e_0 - e_1 \\ &= \frac{q}{2}(\frac{5}{4} - \frac{1}{2}(m_0 + m_1)) - e_0 - e_1 \end{aligned}$$

由于 $\frac{5}{4} - \frac{1}{2}(m_0 + m_1) = \overline{m_0 m_1} \pm \frac{1}{4}$ ，因此有：

$$b - as = \frac{q}{2}\overline{m_0 m_1} \pm \frac{q}{8} - e_0 - e_1 = \frac{q}{2}\overline{m_0 m_1} + e = Enc(\overline{m_0 m_1})$$

由于 $e = \pm \frac{q}{8} - e_0 - e_1$ ，且 $|e_0|, |e_1| < \frac{q}{16}$ 则 $|e| < \frac{q}{4}$ ，因此通过与非同态计算正确得到了 $LWE_s^{2/q}(\overline{m_0 m_1}, \frac{q}{4})$ 。

但是仅仅这样还无法实现全同态运算，因为上述运算得到的结果并非为 $LWE_s^{4/q}(\overline{m_0 m_1}, \frac{q}{16})$ 。FHEW 方案设计了一个**累加器**，并通过**RGSW 方案**进行实例化来完成这个转换任务。

累加器（ACC）

在 FHEW 使用的累加器中，输入和输出均为 LWE 加密方案，在中间过程中会使用**另一个加密方案 E** 。累加器（ACC）包括三个功能：

1. **初始化**：记 v 为初始值，则累加器初始化操作记为 $ACC \leftarrow v$ 。
2. **累加**：记 v 为待累加值，则累加操作（同态执行，需要对数据加密）记为 $ACC \stackrel{+}{\leftarrow} E(v)$ 。
3. **提取最高位**：即同态的提取累加器中加密元素的最高位，结果仍为密文，记为 $c \leftarrow msbExtract(ACC)$ 。

设 $m = \overline{m_0 m_1}$ ，则使用上述三个功能，将 $c = (a, b) = LWE_s^{2/q}(m, \frac{q}{4})$ 转换为 $c' = (a', b') = LWE_s^{4/q}(m, \frac{q}{16})$ 的步骤如下：

1. 初始化 $ACC \leftarrow b + \frac{q}{4}$ 。
2. 对向量 a 中元素进行**分解**，得到 $BitDecomp(a) = (a_{1,0}, \dots, a_{1,d_g-1}, a_{2,0}, \dots, a_{2,d_g-1}, \dots, a_{n,d_g-1})$ 。
3. 对于 $1 \leq i \leq n$ 且 $0 \leq j < d_g$ ，计算 $K_{i,j} = -a_{i,j}s_iB_g^j$ ，并执行**累加操作** $ACC \xleftarrow{+} E(K_{i,j})$ 。
4. 输出 $c' \leftarrow msbExtract(ACC)$ 。

容易验证 ACC 在所有累加结束后（执行 $msbExtract$ 之前）的内容为：

$$v = b + \frac{q}{4} - \sum_{i,j} a_{i,j}s_iB_g^j = b - as + \frac{q}{4} = \frac{q}{2}m + e + \frac{q}{4}$$

由于 $|e| < \frac{q}{4}$ ，显然有 $msbExtract(v) = m$ 成立。可以看出，上述步骤本质上是一种**同态解密**，在另一种加密环境下解密输入的密文，得到另一个环境的密文。即上述步骤全部执行完毕后将得到对 m 的另一种加密，其噪音上限与模数取决于如何实现上述步骤。

RGSW 方案实例化累加器

FHEW 方案采用 RGSW 加密实现上述累加器的功能，首先设置一些参数。

设 $R_Q = \mathbb{Z}_Q[x]/(x^N + 1)$ 为多项式环，其中 $Q = B_g^{d_g}$ ， B_g 为**分解基底**并设为 3 的幂次， N 设为 2 的幂次且要求满足 $q|2N$ （这里为了简单可以直接设 $q = 2N$ ）。

设 t 为最终转换后要求的**明文空间大小**（此处 $t = 4$ ），取**可逆**元素 $u \in \mathbb{Z}_Q$ 为 $\lfloor \frac{Q}{2t} \rfloor$ 和 $\lceil \frac{Q}{2t} \rceil$ 二者之一，由于 Q 为 **3 的幂次**，显然二者必有一个是可逆的。

设消息 $m \in \mathbb{Z}_q$ 被编码为 $y^m \in R_Q$ ，其中 $y = x^{\frac{2N}{q}}$ 。当 $q = 2N$ 时，编码等于 x^m 。

累加器中采用的加密方案 E 设为 **RGSW 加密方案**（缩放因子为 u ，密钥设为 z ，可参考前文介绍如何加密）。设重组矩阵 $G \in R_Q^{2d_g \times 2}$ （这里与原论文顺序不一样，本质上不变）：

$$G = \begin{bmatrix} B_g^0 & 0 \\ B_g^1 & 0 \\ \vdots & \vdots \\ B_g^{d_g-1} & 0 \\ 0 & B_g^0 \\ \vdots & \vdots \\ 0 & B_g^{d_g-1} \end{bmatrix}$$

则初始化操作 $ACC \leftarrow v$ 设为：

$$ACC = ux^vG \in R_Q^{2d_g \times 2}$$

注意到初始化可以看作对 x^v 的一种**特殊 RGSW 加密**（没有LWE加密对），可以理解为**明文版 RGSW**，没有加密特性但满足**同态特性**，也和正常加密一样记为 $RGSW(x^v)$ 。

设 v 为当前累加器内的值（包裹在 RGSW 中），累加操作 $ACC \xleftarrow{+} E(v')$ 设为 RGSW **内积**：

$$ACC = ACC \times E(v') = RGSW(x^v) \times RGSW(x^{v'}) = RGSW(x^{v+v'})$$

显然当累加器的所有累加结束后（执行 $msbExtract$ 之前），容器内为 $RGSW(x^{\frac{q}{2}m+e+\frac{q}{4}})$ 。

设一个**测试向量** $\vec{t} = -\sum_{i=0}^{i<\frac{q}{2}} \vec{y}^i$ ，其中 \vec{y}^i 为多项式 y^i 的长度为 N 的**系数向量**。显然当 $q = 2N$ 时，有 $\vec{t} = -\sum_{i=0}^{i<\frac{q}{2}} \vec{x}^i = \{-1\}^N$ 。该测试向量就是**提取最高位**的关键部分，不难看出 $\vec{t} \cdot \vec{y}^v$ 的值在 $0 \leq v < N$ 时为 -1 ，在 $N \leq v < 2N$ 时为 1 ，这就相当于提取了 v 的最高位。

FHEW 就是利用这个特性来实现累加器的**同态提取最高位**操作 $c \leftarrow msbExtract(ACC)$ ：

首先取累加器容器中的 $RGSW(x^{\frac{q}{2}m+e+\frac{q}{4}})$ 的第 $d_g + 1$ 行（原论文中为第 2 行，**仅为顺序区别**）的值为 $RLWE(x^{\frac{q}{2}m+e+\frac{q}{4}}) = (a, b)$ ，计算：

$$c = (\vec{t} \cdot a, \vec{t} \cdot b + u) = (\vec{t} \cdot a, \vec{t} \cdot az + \vec{t}e + u\vec{t}x^{\frac{q}{2}m+e+\frac{q}{4}} + u)$$

其中 \vec{t} 为**向量**，其与向量乘法为内积运算，结果为**数值**；与多项式乘法可以看作将向量作为多项式系数**组成多项式**后，进行**多项式乘法**，再取**系数向量**，结果还是向量。记 $a' = \vec{t} \cdot a, e' = \vec{t}e$ ，同时有 $u\vec{t}x^{\frac{q}{2}m+e+\frac{q}{4}} + u = 2u \cdot msbExtract(\frac{q}{2}m + e + \frac{q}{4}) = 2um$ ，因此有：

$$c = (a', b') = (a', a'z + e' + 2um)$$

考虑到 $2u \approx \frac{Q}{t}$ ，因此上述密文 c 相当于 $LWE_z^{t/Q}(m)$ 。

随后执行**密钥转换**得到（和 BGV / BFV 方案中的密钥转换类似，本质为**用新密钥加密旧密钥**，这里不再赘述）：

$$c' = LWE_s^{t/Q}(m) = KeySwitch(LWE_z^{t/Q}(m))$$

再执行**模数转换**得到（参考 BGV 方案）：

$$c'' = LWE_s^{t/q}(m) = ModSwitch(LWE_s^{t/Q}(m))$$

至此为累加器的**同态提取最高位**操作 $c \leftarrow msbExtract(ACC)$ 的全部流程，结果为 c'' 。

论文给出了 c'' 噪声大小 $|e''| < \frac{q}{16}$ 的详细证明，包括如何合理的选取参数，这里不再展开详细讨论，只进行简单说明。可以看出原密文的噪声上限 $\frac{q}{4}$ 在经过**最高位提取之后被消除了**（因为噪声在 x^v 的**指数 v** 中，最高位提取结果要么为 0 要么为 1），而新产生的噪声只与累加器中的 RGSW **加密和乘法运算** 有关。而 RGSW 加密时引入的噪声可以根据安全参数合理设置，密文同态内积产生的噪声为若干 $B_g e$ 之和，通过合理的设置参数就能实现最终的密文噪声大小满足 $|e''| < \frac{q}{16}$ 。

可以看到，经过上述累加器运算，FHEW 方案将 XAND 运算的输出 $LWE_s^{2/q}(\overline{m_0 m_1}, \frac{q}{4})$ 转换为了 XAND 运算合法的输入 $LWE_s^{4/q}(\overline{m_0 m_1}, \frac{q}{16})$ ，等价于实现了 XAND 电路门的 Bootstrapping 运算，这意味着 FHEW 方案支持**任意深度**布尔电路的全同态运算，每经过一个电路门执行一次 Bootstrapping。

10. TFHE 加密方案

TFHE 方案由 Chillotti、Gama、Georgieva 和 Izabachène 发表^[16]，该方案优化了 FHEW 中的 Bootstrapping 操作，并给出一种**更通用**的 LWE 和 GSW 定义以及若干同态计算方法。

该方案中引入了一个**新的数据结构** Torus，在该结构上加密数据或多项式的 LWE 和 GSW 算法分别记为 T(R)LWE 和 T(R)GSW。引入 Torus 主要目的是对若干类似加密方案进行**统一表示**，可以衍生出如 LWE、RLWE、approx-GCD 等加密方式。为了便于理解，在介绍 TFHE 方案时仍然沿用前文的表示方式（LWE、RLWE、GSW、RGSW）。

同态逻辑计算

TFHE 方案给出了若干 LWE-to-LWE 的**同态逻辑计算门**（NOT、AND、NAND、OR、XOR），其主要思想与 FHEW 一致，均为利用一些线性变换将逻辑运算转换为**同态加法和最高位的提取**，且加密方式与 FHEW 类似，线性变换简单描述如下：

- 同态 NOT: $(0, \frac{1}{4}) - c$
- 同态 AND: $(0, -\frac{1}{8}) + c_1 + c_2$
- 同态 NAND: $(0, \frac{5}{8}) - c_1 - c_2$
- 同态 OR: $(0, \frac{1}{8}) + c_1 + c_2$
- 同态 XOR: $2 \cdot (c_1 - c_2)$

Bootstrapping 优化

TFHE 方案给出了一个 CMux 门来替代 FHEW 方案在 Bootstrapping 时使用的 RGSW 的内积，其中 CMux 门由 RGSW 和 RLWE 的**外积**组成，因此能省去 FHEW 中 RGSW 内积浪费的时间和空间。

CMux 选择门

CMux 选择门由两个**输入** d_0, d_1 和一个**控制端** C 组成，其中 C 为由 RGSW 方案加密的单个比特 $c \in \{0, 1\}$ ，而 d_0, d_1 为由 RLWE 方案加密的数据。

该门的作用是根据控制信号选择输出：当 $c = 0$ 时输出 d_0 ，否则输出 d_1 。实现方式如下：

$$CMux(C, d_0, d_1) = C \square (d_1 - d_0) + d_0$$

显然上述计算由一次 RGSW 和 RLWE 的**内积运算**与若干 RLWE 的**同态加法**同态计算完成，正确性显然可证。

TFHE 累加器

TFHE 方案在 Bootstrapping 中使用的累加器 (ACC) 本质上与 FHEW 方案一致。

在 TFHE 方案中，在对多项式乘上 x^v 称作“**旋转**” (因为仅多项式系数顺序和符号改变)，并利用一个旋转多项式 $1 + x + x^2 + \dots + x^{N-1}$ 来代替 FHEW 方案中的测试向量 \vec{t} ，但两者本质都是根据指数的不同取值影响来提取指数的最高位。

同时，TFHE 方案将累加器的累加操作表示为一个 CMux 选择门。考虑到 FHEW 方案在进行累加时，待累加元素为 $a_{i,j} s_i B_g^j$ ，TFHE 方案将累加操作转换为：

$$ACC \leftarrow CMux(RGSW(s_i), x^{a_i} \cdot ACC, ACC)$$

其中 Bootstrapping 的输入 LWE 密文加密的密钥为 $s \in \{0, 1\}^n$ ，容易验证：

$$CMux(RGSW(s_i), x^{a_i} \cdot ACC, ACC) = x^{a_i s_i} \cdot ACC$$

和 FHEW 方案要实现的目的一样，但是**从内积变成了外积**，时间和空间都有所优化。可以注意到在 FHEW 方案中 Bootstrapping 最终只需要取结果矩阵的**某一行**即可，因此会有大量的**冗余计算**，TFHE 正是对此进行了优化。

TFHE 密钥转换

TFHE 方案中给出了两种密钥转换算法：Public Functional Key Switching 和 Private Functional Key Switching，两者均可把 p 个 LWE 加密密文转换为另一种指定密钥的 LWE 或 RLWE 密文，且转换过程中可以对加密的密文执行一次**函数运算** $f: \mathbb{Z}^p \rightarrow \mathbb{Z}[x]$ (即把**整数向量**以某种规则**转换为一个多项式**)。

Public Functional Key Switching 和 Private Functional Key Switching 的区别是前者函数 f 是**公开的**，而后者是**不公开的**，转换思路与正常的密钥转换类似，只是在**转换时额外计算**函数 f 。

在 TFHE 的 Gate-Bootstrapping (LWE to LWE) 中最后还是使用正常的密钥转换 (与 FHEW 一致) 进行处理，设计上述密钥转换是为了实现 TFHE 的 Circuit-Bootstrapping (LWE to RGSW)。因为 TFHE 提供了很多输入 RGSW 密文输出 LWE 密文的同态计算电路，为了实现**无限深度**的全同态计算必须提供对应的 Bootstrapping 运算，感兴趣可以查看原文。

至此，对于同态加密技术的大致介绍结束。事实上还有很多没有提到的内容，或许会在以后进行补充。

参考文献 (续)

- [14] C. Gentry, A. Sahai, and B. Waters, “Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based,” in Advances in Cryptology – CRYPTO 2013, R. Canetti and J. A. Garay, Eds., Berlin, Heidelberg: Springer, 2013, pp. 75–92.
- [15] L. Ducas and D. Micciancio, “FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second,” in Advances in Cryptology – EUROCRYPT 2015, E. Oswald and M. Fischlin, Eds., Berlin, Heidelberg: Springer, 2015, pp. 617–640.

- [16] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast Fully Homomorphic Encryption Over the Torus," J. Cryptol., vol. 33, no. 1, pp. 34–91, Jan. 2020.
-

本文为作者在学习相关知识时的一种记录，便于以后的回顾。作者并没有系统地学习过密码学，因此在表述上可能会存在不严谨甚至出错的地方，文章仅供参考，欢迎大家与我交流，一起进步！

其他平台：

- 知乎 (Totoro)： <https://www.zhihu.com/people/totoro-14-60>
- CSDN (_Totoro_)： https://blog.csdn.net/orz_Totoro
- B站 (Totoro_134)： <https://space.bilibili.com/279377771>
- Github (Totoro134)： <https://github.com/Totoro134>
- 公众号 (知识长生所)