



同态加密方案

Homomorphic Encryption Scheme

Yilun Sheng

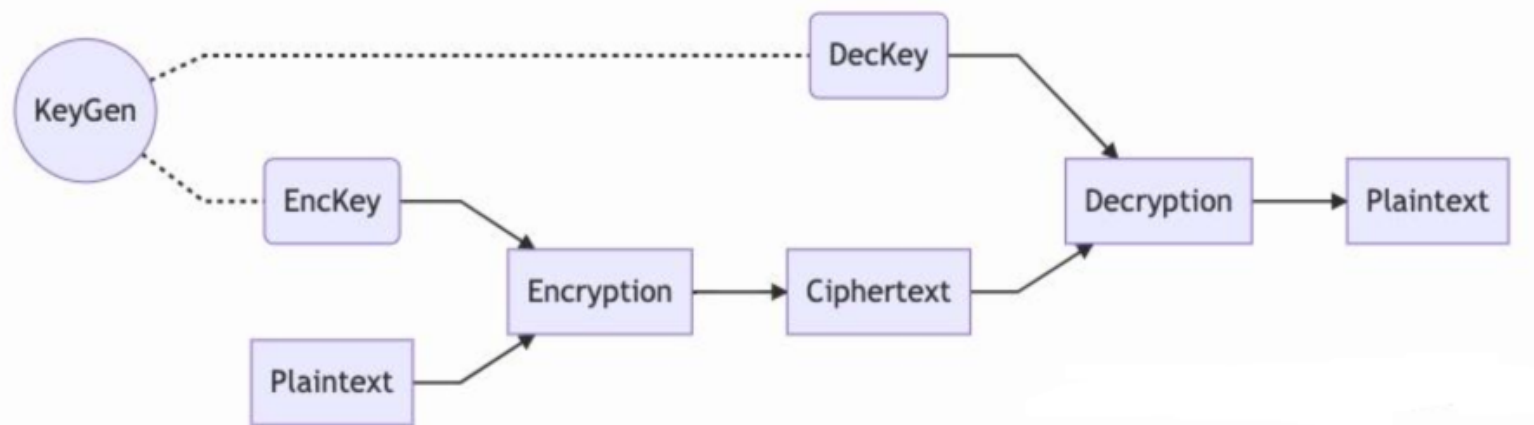
October 13



Encryption Scheme

所有的加密体系，无疑是做了三件事：

- (1) 通过密钥生成算法 $KeyGeneration(1^\lambda)$ 随机生成一对用于加密和解密的密钥($EncKey, DecKey$)；
- (2) 加密方通过加密密钥 $EncKey$ 和加密算法 $Encryption$ 加密原文 $PlainText$ ，得到密文 $CipherText$ ；
- (3) 解密方通过解密密钥 $DecKey$ 和解密算法 $Decryption$ 解密密文 $CipherText$ ，还原原文 $PlainText$ 。



一般来说，加密体系分为两种：

- (1) 对称加密体系：加密和解密使用的密钥相同。
- (2) 非对称加密体系：使用公钥 $Public\ Key$ 加密，使用私钥 $Private\ Key$ 解密。

Homomorphic Encryption Scheme

Definition 1. Homomorphism is defined as a map preserving all the algebraic structures between the domain and range of an algebraic set.

Definition 2. An encryption scheme is called homomorphic over an operation ‘ $$ ’ if it supports the following equation:*

$$E(m_1) * E(m_2) = E(m_1 * m_2), \forall m_1, m_2 \in M$$

where E is the encryption algorithm and M is the set of all possible messages.

Homomorphic over Addition

具有加法同态属性的加密算法：

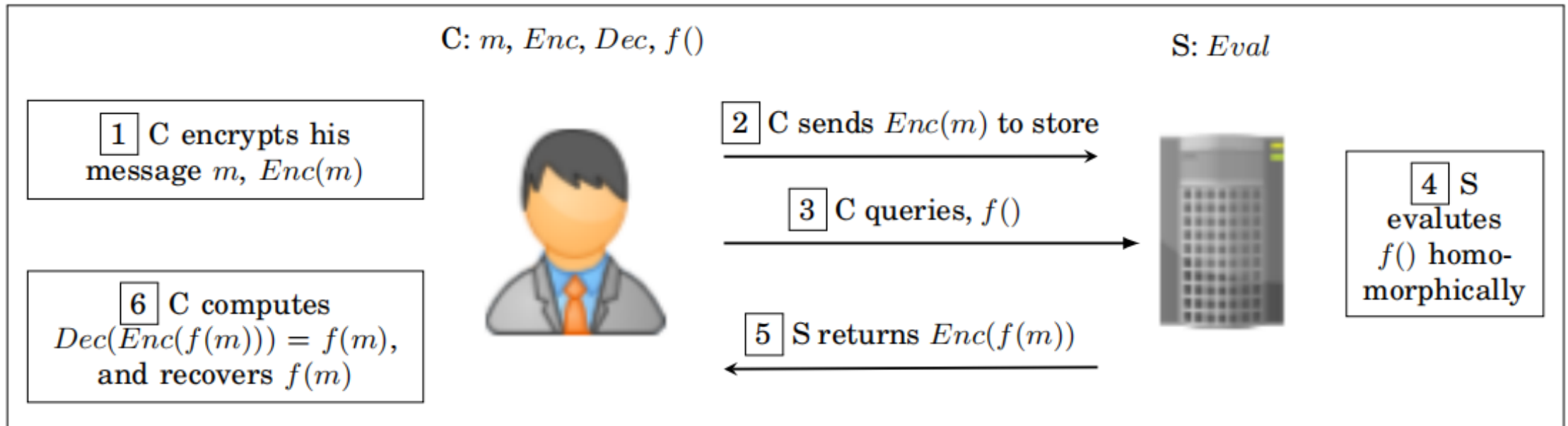
$$Enc(1) = ct_1, Enc(2) = ct_2$$

$$Enc(3) = Enc(1 + 2) = ct_1 + ct_2$$

我们可以使用加法同态的加密算法：

- (1) 通过密钥生成算法 *KeyGeneration* 生成公私钥对 (pk, sk) ，公钥 pk 分发给每个员工
- (2) 每个员工将自己的选择 $pt_i \in 0, 1$ 加密后发送给服务器，即 $ct_i = Encryption(pk, pt_i)$
- (3) 将每个员工的选择累加后发送给大老板，用私钥 sk 解密，即 $Decryption(sk, \sum_i ct_i) = \sum_i pt_i$

A Simple Client-Server HE Scenario



PHE schemes are developed in some applications like e – voting or Private Information Retrieval where C is Client and S is Server.

Propaedeutics Related to RSA Algorithm

Definition 3. 欧拉函数: m 个整数 $1, 2, \dots, m$ 中与 m 互素的整数的个数, 记为 $\varphi(m)$.

Theorem 1. 若 p 为素数, 则 $\varphi(p) = p - 1$.

Theorem 2. 对于素数幂 $m = p^\alpha$, 有 $\varphi(m) = p^\alpha - p^{\alpha-1}$.

Theorem 3. 若 p, q 为两个不同的素数, 则 $\varphi(pq) = (p - 1)(q - 1)$.

Theorem 4. (Euler) 设 m 是大于 1 的整数, 且 $(a, m) = 1$, 则 $a^{\varphi(m)} \equiv 1 \pmod{m}$.

Homomorphic over Multiplication — RSA

密钥生成算法 *KeyGeneration* :

- (1) 选择两个不同的大素数 p, q .
- (2) 计算 $n = p \times q$, $\varphi(n) = (p-1)(q-1)$.
- (3) 选择整数 e 使得 $(\varphi(n), e) = 1$, $1 < e < \varphi(n)$
- (4) 计算整数 d 使得 $d = e^{-1} \bmod \varphi(n)$.
- (5) 得到公钥 $\{e, n\}$ 和私钥 $\{d, n\}$.

加密算法 *Encryption* :

$$c = m^e \bmod n.$$

解密算法 *Decryption* :

$$m = c^d \bmod n.$$

Homomorphic Property: for $m_1, m_2 \in M$

$$E(m_1) * E(m_2) = (m_1^e \bmod n) * (m_2^e \bmod n) = (m_1 * m_2)^e \bmod n = E(m_1 * m_2)$$

密钥生成算法 *KeyGeneration* :

- (1) $p = 17$, $q = 19$.
- (2) $n = 17 \times 19 = 323$, $\varphi(n) = 16 \times 18 = 288$.
- (3) $(288, 5) = 1$, $e = 5$.
- (4) $d = 5^{-1} \bmod 288 = 29$.
- (5) 得到公钥 $\{5, 323\}$ 和私钥 $\{29, 323\}$.

加密算法 *Encryption* :

$$c = 2^5 = 32 \bmod 323.$$

解密算法 *Decryption* :

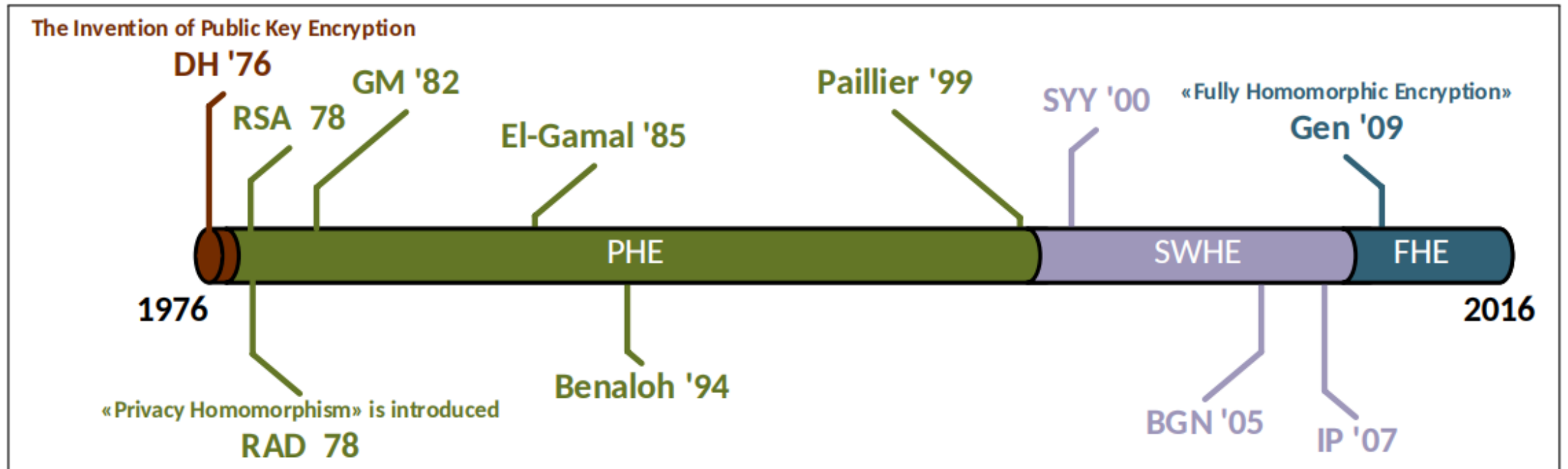
$$m = 32^{29} = 2^{128+16+1} = 137 \times 290 \times 2 = 2 \bmod 323.$$

Categories of HE

All different HE attempts can neatly be categorized under three types of schemes with respect to the number of allowed operations on the encrypted data as follows:

<i>Categories</i>	<i>Properties</i>
Partially Homomorphic Encryption (PHE)	allows only one type of operation with an unlimited number of times
Somewhat Homomorphic Encryption (SWHE)	allows some types of operations with a limited number of times
Fully Homomorphic Encryption (FHE)	allows an unlimited number of operations with an unlimited number of times

Timeline of HE Schemes



Gentry's FHE scheme is the first plausible and achievable FHE scheme. It is based on ideal—lattices in math and it is not only a description of the scheme, but also a powerful framework for achieving FHE.



CKKS 全同态加密算法

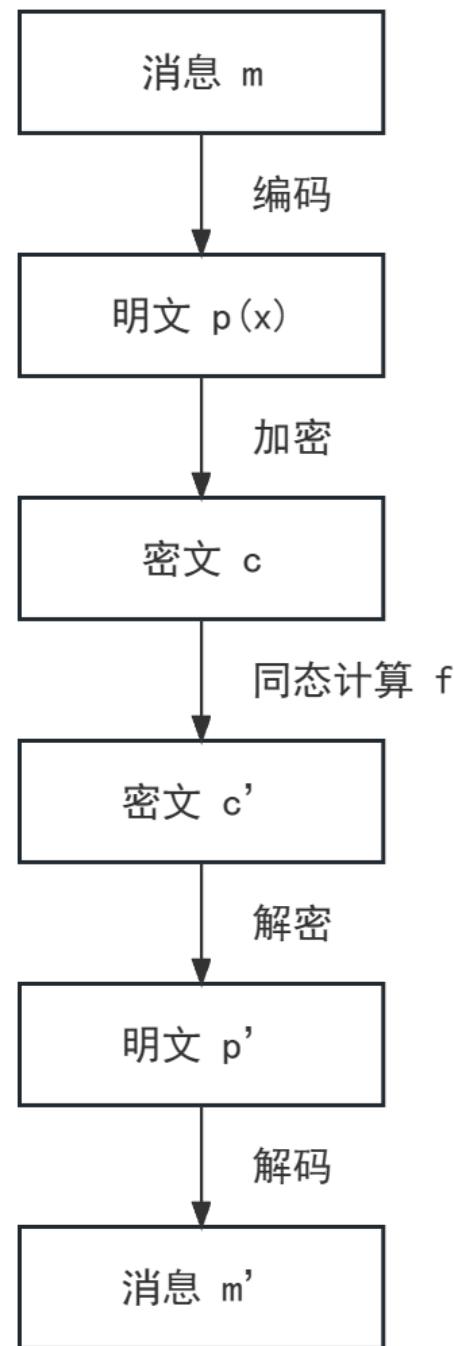
CKKS Fully Homomorphic Encryption Algorithm



Algorithms in CKKS

CKKS 算法是论文 *Homomorphic Encryption for Arithmetic of Approximate Numbers* 中提出的近似计算同态加密算法，其具体构造方法基于 BGV 算法，也可以基于其他类型的全同态加密算法进行构造。它是目前对于实数和复数进行近似同态计算最为有效的方法，已被广泛用于机器学习等领域。

CKKS 全同态加密能够对实数和复数向量进行同态计算，从消息编码到明文，到加密成密文，以及解密和解码成消息，整个流程在各个数字对象上转换，如右图所示。





格密码理论基础

Basic Theory of Lattice Cryptography



Propaedeutics Related to Encoding & Decoding

在 CKKS 方案中，明文空间在复数域上。基于 RLWE 问题，计算是在环（整数多项式）上，所以在加密之前需要进行编码，将一个复数向量编码成一个整数多项式，然后再进行加密和计算。

LWE 搜索问题：给出一些关于秘密向量 \mathbf{s} 的近似随机线性方程，其目的是恢复秘密向量 \mathbf{s} ，例如：

$$\left\{ \begin{array}{l} 14s_1 + 15s_2 + 5s_3 + 2s_4 \approx 8(mod\ 17) \\ 13s_1 + 14s_2 + 14s_3 + 6s_4 \approx 16(mod\ 17) \\ 6s_1 + 10s_2 + 13s_3 + 1s_4 \approx 3(mod\ 17) \\ 10s_1 + 4s_2 + 12s_3 + 16s_4 \approx 12(mod\ 17) \\ \dots\dots\dots \\ 9s_1 + 5s_2 + 9s_3 + 6s_4 \approx 9(mod\ 17) \end{array} \right.$$

在上述每个方程中加入一个小的错误，例如错误取自 $-1 \sim +1$ ，目标是恢复向量 \mathbf{s} 。若上述方程中没有加入错误，使用高斯消元法就能在多项式时间内恢复向量 \mathbf{s} ，但是加入错误后问题变得非常困难。

LWE Examples

2	13	7	3
4	7	9	1
6	14	5	11

 \cdot

8
3
12
5

 $+$

1
-1
2

 $=$

13
12
3

n 维

A {

a_1
 a_2
 \dots
 a_m

+

s

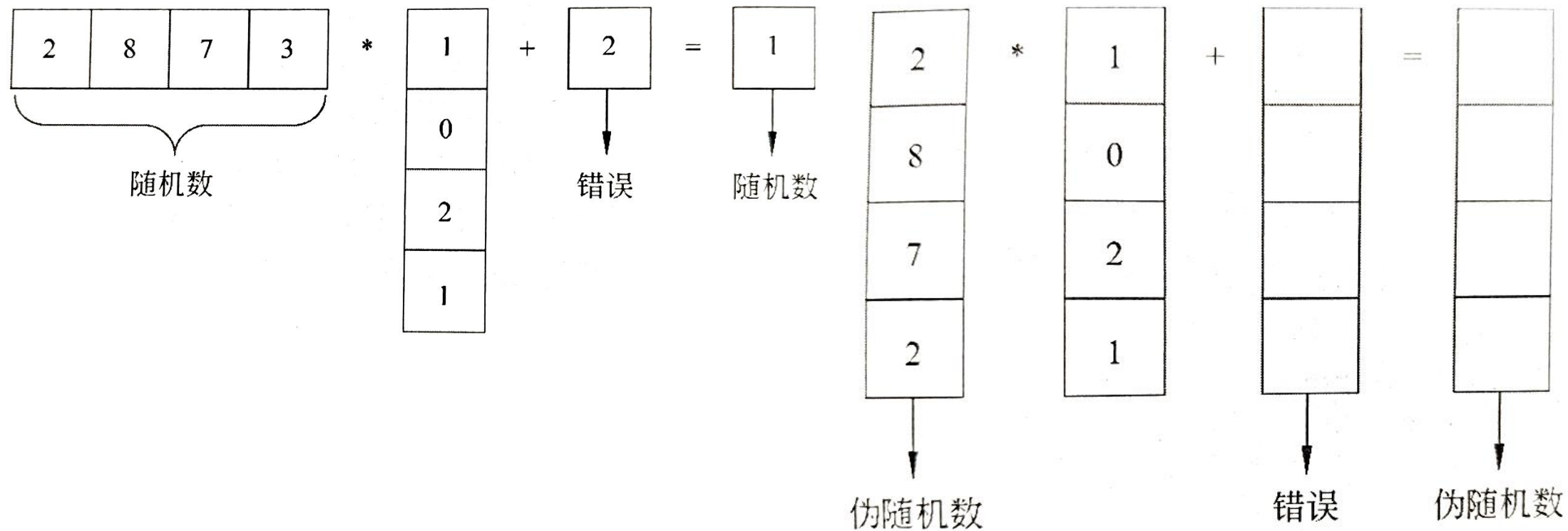
+

e

=

b

Ring LWE Examples



Polynomial Ring

例： 设 $p(x) = x^8 + x^4 + x^3 + x + 1$ 是 $F_2[X]$ 中的 8 次不可约多项式

则对于 $f(x) = x^6 + x^3 + x + 1$, $g(x) = x^5 + x^2 + x + 1$, 有

$$f(x) + g(x) = x^6 + x^5 + x^3 + x^2 \pmod{p(x)}$$

$$f(x) \cdot g(x) = x^6 + x^3 + 1 \pmod{p(x)}$$

Propaedeutics Related to Encoding & Decoding

在 CKKS 方案中，明文空间在复数域上。基于 RLWE 问题，计算是在环（整数多项式）上，所以在加密之前需要进行编码，将一个复数向量编码成一个整数多项式，然后再进行加密和计算。

Definition 4. 令 ξ 是 $2N$ 次单位原根，即 $\xi^{2N} = 1 \bmod t$ 且不存在正整数 $x < 2N$ 使得 $\xi^x = 1 \bmod t$

Theorem 5. $x^N + 1 = \prod_{i=0}^{N-1} (x - \xi^{2i+1})$

Theorem 6. $R_t = \frac{Z_t[x]}{x^N + 1} = \frac{Z_t[x]}{\prod_{i=0}^{N-1} (x - \xi^{2i+1})} \approx \prod_{i=0}^{N-1} \frac{Z_t[x]}{x - \xi^{2i+1}} = \prod_{i=0}^{N-1} Z_t[\xi^{2i+1}]$

Encoding & Decoding — BGV

将一个复数向量 $\mathbf{z} \in C^N$ 编码成一个多项式 $p(x) \in C[X]/(X^N + 1)$, 需要计算 σ^{-1}

给一个复数向量 $\mathbf{z} \in C^N$ 去找到一个多项式 $p(x) = \sum_{i=0}^{N-1} p_i X^i \in C[X]/(X^N + 1)$, 使得

$$\sigma(p(x)) = (p(\xi^1), p(\xi^3), \dots, p(\xi^{2^{N-1}})) = (z_0, z_1, \dots, z_{N-1})$$

也就是求解关于 $(p_0, p_1, \dots, p_{N-1})$ 的线性方程组

$$\begin{cases} p_0 \xi^{1 \cdot 0} + p_1 \xi^{1 \cdot 1} + \dots + p_{N-1} \xi^{1 \cdot (N-1)} = z_0 \\ p_0 \xi^{3 \cdot 0} + p_1 \xi^{3 \cdot 1} + \dots + p_{N-1} \xi^{3 \cdot (N-1)} = z_1 \\ \dots \dots \dots \\ p_0 \xi^{(2^{N-1}) \cdot 0} + p_1 \xi^{(2^{N-1}) \cdot 1} + \dots + p_{N-1} \xi^{(2^{N-1}) \cdot (N-1)} = z_{N-1} \end{cases}$$

记 $A \cdot p = \mathbf{z}$, 其中 A 是关于 $(\xi^{2^{i-1}})_{i=1,2,\dots,N}$ 的范德蒙矩阵, p 是多项式的系数, \mathbf{z} 是需要编码的向量

Encoding & Decoding — BGV

因此，已知 z 和 ξ 就可以将 z 编码为一个 $C[X]/(X^N + 1)$ 上的多项式：

$$\begin{bmatrix} p_0 \\ \vdots \\ p_{N-1} \end{bmatrix} = \begin{bmatrix} \xi^{1 \cdot 0} & \dots & \xi^{1 \cdot (N-1)} \\ \vdots & \vdots & \vdots \\ \xi^{(2N-1) \cdot 0} & \dots & \xi^{(2N-1) \cdot (N-1)} \end{bmatrix}^{-1} \cdot \begin{bmatrix} m_0 \\ \vdots \\ m_{N-1} \end{bmatrix}$$

解码过程也很简单：

$$\begin{bmatrix} m_0 \\ \vdots \\ m_{N-1} \end{bmatrix} = \begin{bmatrix} \xi^{1 \cdot 0} & \dots & \xi^{1 \cdot (N-1)} \\ \vdots & \vdots & \vdots \\ \xi^{(2N-1) \cdot 0} & \dots & \xi^{(2N-1) \cdot (N-1)} \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ \vdots \\ p_{N-1} \end{bmatrix}$$

Encoding & Decoding Example — BGV

例如：令 $M=8$ ， $N=4$ ，则 $\Phi_M(X) = X^4 + 1$ ，选取 $\xi = e^{i\pi/4} = 0.7 + 0.7i$ ，对 $(1, 2, 3, 4)$ 进行编码：

$$\begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} 1 & \xi^1 & \xi^2 & \xi^3 \\ 1 & \xi^3 & \xi^6 & \xi^9 \\ 1 & \xi^5 & \xi^{10} & \xi^{15} \\ 1 & \xi^7 & \xi^{14} & \xi^{21} \end{bmatrix}^{-1} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 2.5 + 4.4 \times 10^{-16}i \\ -5.0 \times 10^{-16} + 0.7i \\ -3.5 \times 10^{-16} + 0.5i \\ -8.3 \times 10^{-16} + 0.7i \end{bmatrix}$$

解码后的向量为：

$$\begin{bmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} 1 & \xi^1 & \xi^2 & \xi^3 \\ 1 & \xi^3 & \xi^6 & \xi^9 \\ 1 & \xi^5 & \xi^{10} & \xi^{15} \\ 1 & \xi^7 & \xi^{14} & \xi^{21} \end{bmatrix} \cdot \begin{bmatrix} 2.5 + 4.4 \times 10^{-16}i \\ -5.0 \times 10^{-16} + 0.7i \\ -3.5 \times 10^{-16} + 0.5i \\ -8.3 \times 10^{-16} + 0.7i \end{bmatrix} = \begin{bmatrix} 1 - 1.1 \times 10^{-16}i \\ 2 - 4.7 \times 10^{-16}i \\ 3 + 2.8 \times 10^{-17}i \\ 4 + 2.2 \times 10^{-16}i \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

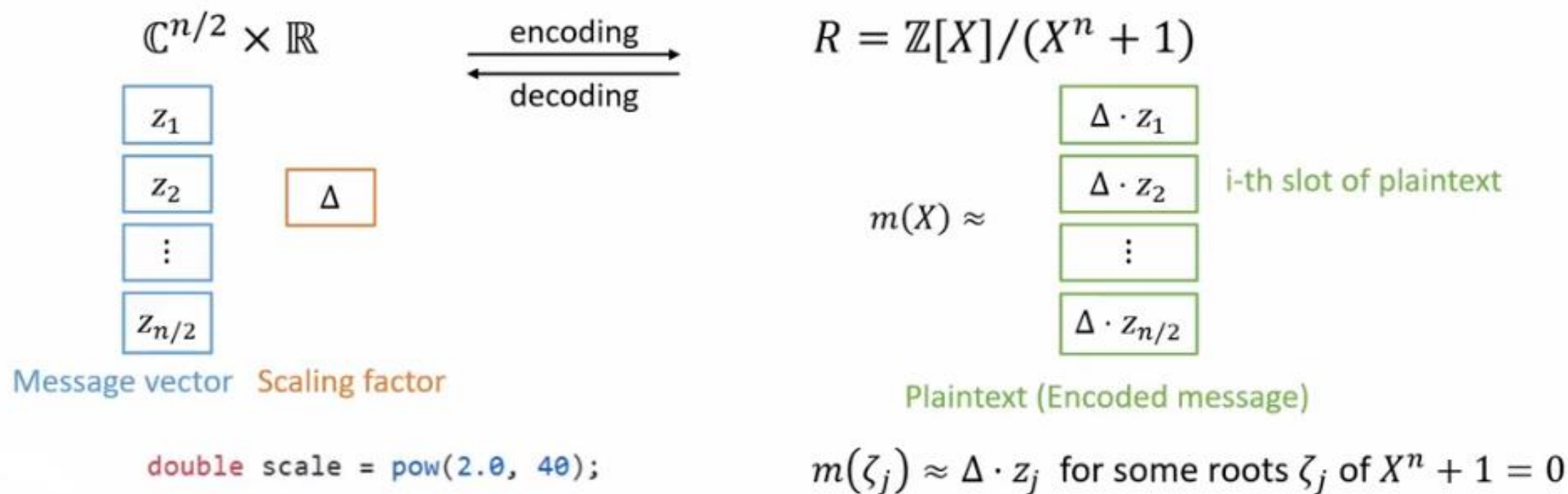
Encoding & Decoding — CKKS

如果在 CKKS 算法中使用 $N/2$ 空间中的复数向量，就需要复制另外一半共轭根进行扩展。在 CKKS 算法中定义了一个操作 π ，该操作将 $\sigma(R) \subseteq H$ 中的元素投影到 $C^{N/2}$ 中，并且该投影是同构的。因此，对于 $z \in C^{N/2}$ ，要想将其编码为 R 上的多项式，首先使用 π^{-1} 将 z 扩展到 H 上，即 $\pi^{-1}(z) \in H$ 。

但是 H 中的元素未必落在 $\sigma(R)$ 上，因此只有通过同构映射将 $\pi^{-1}(z) \in H$ 映射到 $\sigma(R)$ 上，才能使用 $\sigma^{-1}(R)$ 将 z 编码为 R 上的整系数多项式。我们这里使用论文 *A Toolkit for Ring - LWE Cryptography* 中提出的一个技巧：坐标随机取整，该方法允许将实数 x 取整到 $\lfloor x \rfloor$ 或 $\lfloor x \rfloor + 1$ ，距离越近概率越高。

另外需要注意的是，坐标随机取整时，可能会影响一些有效数字。为了保证有效数字的精度，例如保证 $1/\Delta$ 的精度，编码时乘以 Δ ，解码时除以 Δ 。例如：当对 $x = 1.4$ 随机取整时，则可以令 $\Delta = 4$ ，保留精度 $1/\Delta = 0.25$ ，有 $\Delta \cdot x = 5.6 \approx 6$ ，而 $6/\Delta = 1.5$ ，从而保证了 0.25 范围内的精度。

Encoding & Decoding — CKKS



Encoding & Decoding Example — CKKS

总结一下 CKKS 算法的编码过程：

- (1) 假设向量 $\mathbf{z} \in C^{N/2}$ ，目标是将其编码为 $R = Z[X]/(X^N + 1)$ 上的一个整系多项式。
- (2) 利用 $\pi^{-1}(\mathbf{z}) \in H$ ，将其扩展到 H 上。
- (3) 计算 $\Delta \cdot \pi^{-1}(\mathbf{z})$ ，保证精度。
- (4) 使用坐标随机取整投影到 $\sigma(R)$ 上，即 $[\Delta \cdot \pi^{-1}(\mathbf{z})]_{\sigma(R)} \in \sigma(R)$ 。
- (5) 使用 $\sigma(p(x)) = \sigma^{-1}([\Delta \cdot \pi^{-1}(\mathbf{z})]_{\sigma(R)}) \in R$ 进行编码。

令 $N = 4$ ， $\Delta = 64$ ，那么 $[\xi^1, \xi^3, \xi^5, \xi^7]$ 是 $x^N + 1 = 0$ 的原根，现在执行编码算法：

- (1) $\mathbf{z} = (3 - 4i, 2 + i) \in C^2$
- (2) $\pi^{-1}(\mathbf{z}) = (3 - 4i, 2 + i, 2 - i, 3 + 4i)$
- (3) $\sigma^{-1}(\pi^{-1}(\mathbf{z})) = 1/4 \cdot (10 + 4\sqrt{2}x + 10x^2 + 2\sqrt{2}x^3)$
- (4) $\Delta \cdot \sigma^{-1}(\pi^{-1}(\mathbf{z})) = 16 \cdot (10 + 4\sqrt{2}x + 10x^2 + 2\sqrt{2}x^3)$
- (5) $p(x) = [\Delta \cdot \sigma^{-1}(\pi^{-1}(\mathbf{z}))] = 160 + 91x + 160x^2 + 45x^3$

Algorithms in CKKS — Definition

1 初始化 *CKKS*. *Setup*

p	基
q_0	模数
L	深度上限
q_l	$q_0 \cdot p^l$ ($1 < l \leq L$)
q_L/Q	$q_0 \cdot p^L$
λ	安全等级系数
P	特殊模数（用于重缩放）
$\mathcal{DG}(\sigma^2)$	从 Z^N 中抽取一个 N 维多项式向量，其中每一个系数都是取自方差为 σ^2 的离散高斯分布
$\mathcal{HWT}(h)$	从 $\{0, \pm 1\}^N$ 中抽取一个汉明重量等于 h 的 N 维向量
$\mathcal{ZO}(\rho)$	从 $\{0, \pm 1\}^N$ 中抽取一个 N 维向量，抽取到 1 和 -1 的概率都是 $\rho/2$ ，抽取到 0 的概率是 $1 - \rho$

Algorithms in CKKS — KeyGeneration

2 密钥生成算法 *CKKS. KeyGen*(1^λ)

(1) 输入: λ

(2) 选择: $M = M(\lambda, q_L)$ $h = h(\lambda, q_L)$ $P = P(\lambda, q_L)$ $\sigma = \sigma(\lambda, q_L)$

(3) 抽取: $s \leftarrow \mathcal{HWT}(h)$ $a \leftarrow R_{q_L}$ $e \leftarrow \mathcal{DG}(\sigma^2)$ $b \leftarrow -a \cdot s + e \bmod q_L$
 $a' \leftarrow R_{P \cdot q_L}$ $e' \leftarrow \mathcal{DG}(\sigma^2)$ $b' \leftarrow -a' \cdot s + e' + P \cdot s^2 \bmod P \cdot q_L$

(4) 生成: $sk \leftarrow (1, s)$ $pk \leftarrow (b, a) \in R_{q_L}^2$ $evk \leftarrow (b', a') \in R_{q_L}^2$

(5) 输出: (sk, pk, evk)

3 加密算法 *CKKS. Enc_{pk}*(m)

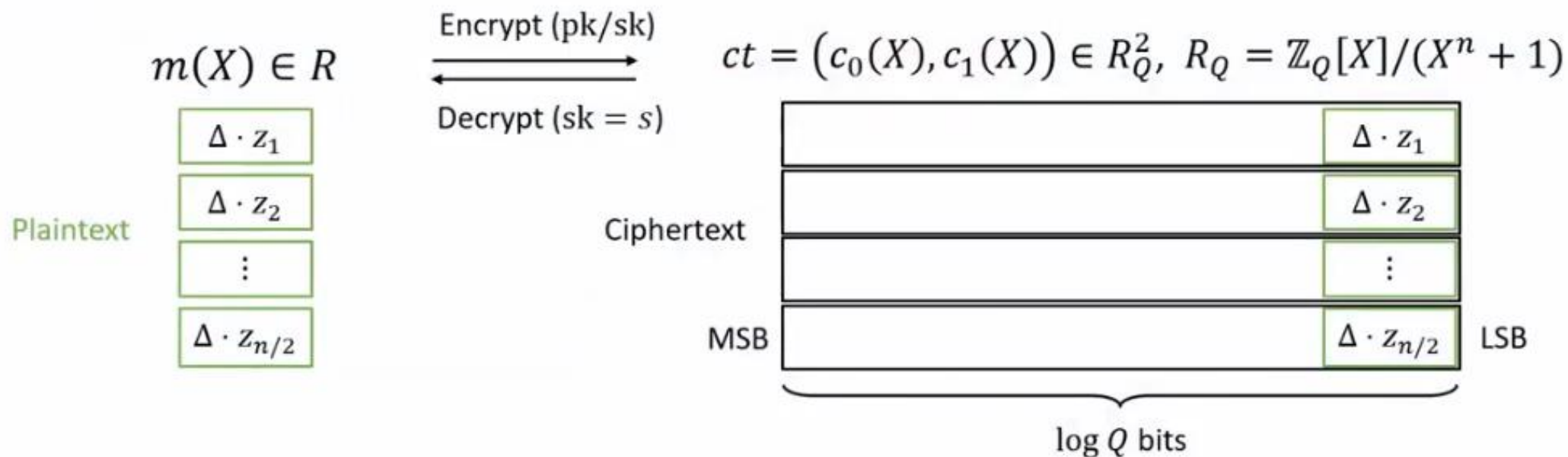
(1) 选取: $v \leftarrow ZP(1/2)$ $e_0, e_1 \leftarrow DG(\sigma^2)$

(2) 输出: $\mathbf{c} = v \cdot pk + (m + e_0, e_1) \bmod q_L$

4 解密算法 *CKKS. Dec_{sk}*(\mathbf{c})

(1) 输出: $m = \langle \mathbf{c} \cdot sk \rangle \bmod q_L$

Algorithms in CKKS — Enc & Dec



- Encrypt: $m(X) \mapsto ct = (c_0(X), c_1(X)) \in R_Q^2$ such that $c_0 + c_1 s \approx m \pmod{Q}$
 - Correctness: $\|m\| < Q$.
 - Notation: $ct(S) = c_0 + c_1 S \in R_Q[S]$
- **Warning:** An encryption of m is not decrypted to exactly m but $m + e$ for some error e such that $|e| < bound$

Algorithms in CKKS — Validity Check

如果是一次加密的明文，检验一下正确性：

$$\begin{aligned}mm &= \langle \mathbf{c} \cdot sk \rangle \bmod q_L \\&= \langle (v \cdot pk + (m + e_0, e_1)), (1, s) \rangle \bmod q_L \\&= \langle (v \cdot b + m + e_0, v \cdot a + e_1), (1, s) \rangle \bmod q_L \\&= \langle (v \cdot (-a \cdot s + e) + m + e_0, v \cdot a + e_1), (1, s) \rangle \bmod q_L \\&= -v \cdot a \cdot s + v \cdot e + m + e_0 + v \cdot a \cdot s + e_1 \cdot s \bmod q_L \\&= v \cdot e + e_0 + m + e_1 \cdot s \bmod q_L \\&= m + (v \cdot e + e_0 + e_1 \cdot s) \bmod q_L \\&\approx m\end{aligned}$$

如果要想误差可以忽略， $v \cdot e + e_0 + e_1 \cdot s$ 应该相对于 q_L 足够小（至于更浅层的模数，误差会随着重缩放变得更小）这些参数在初始化时应该考虑到。

Algorithms in CKKS — Add & Mul

5 同态加法 *CKKS. Add*($\mathbf{c}_1, \mathbf{c}_2$)

对于两个密文 $\mathbf{c}, \mathbf{c}' \in R_{q_l}^2$, 输出 $\mathbf{c}_{Add} \leftarrow \mathbf{c} + \mathbf{c}' \bmod q_l$, $\mathbf{c}_{Add} \in R_{q_l}^2$

解密时的误差是两者各自解密误差的和。

6 同态乘法 *CKKS. Mul*($\mathbf{c}_1, \mathbf{c}_2$)

对于密文 $\mathbf{c} \in R_{q_l}^2$, 明文 $m \in R_{q_l}$, 输出 $\mathbf{c}_{Mul} \leftarrow m \cdot \mathbf{c} \bmod q_l$, $\mathbf{c}_{Mul} \in R_{q_l}^2$

解密时的误差是原先的 m 倍, 仍可忽略。

对于两个密文 $\mathbf{c} = (c_0, c_1)$, $\mathbf{c}' = (c_0', c_1')$, 计算 $(d_0, d_1, d_2) = (c_0 c_0', c_0 c_1' + c_1 c_0', c_1 c_1') \bmod q_l$, 输出 $\mathbf{c}_{Mul} \leftarrow (d_0, d_1) + [P^{-1} \cdot d_2 \cdot evk] \bmod q_l$

Algorithms in CKKS — Relinearlization

先考虑两个解密值的乘积：

$$\begin{aligned} \text{Dec}(c) \cdot \text{Dec}(c') &= (c_0 + c_1 \cdot s) \cdot (c_0' + c_1' \cdot s) \bmod q_l \\ &= c_0 \cdot c_0' + (c_0 \cdot c_1' + c_0' \cdot c_1) \cdot s + c_1 \cdot c_1' \cdot s^2 \bmod q_l \\ &= d_0 + d_1 \cdot s + d_2 \cdot s^2 \bmod q_l \end{aligned}$$

有没有办法将密文的规模保持在 $R_{q_l}^2$ 呢？假设密文只有 (d_0, d_1) ，则解密结果是 $d_0 + d_1 \cdot s \bmod q_l$ ，还缺乏一个 s 的二次项，需要增添一个修补，这里计算辅助密钥 evk 就起到了关键作用。

$$\begin{aligned} \langle P^{-1} \cdot d_2 \cdot evk, sk \rangle \bmod q_l &= P^{-1} \cdot d_2 \cdot b' + P^{-1} \cdot d_2 \cdot a' \cdot s \bmod q_l \\ &= (P^{-1} \cdot d_2 \cdot (-a' \cdot s + e' + P \cdot s^2)) \bmod P \cdot q_L + (P^{-1} \cdot d_2 \cdot a' \cdot s) \bmod q_l \\ &= (P^{-1} \cdot (-d_2 \cdot a' \cdot s + d_2 \cdot e') + d_2 \cdot s^2) \bmod P \cdot q_L + (P^{-1} \cdot d_2 \cdot a' \cdot s) \bmod q_l \\ &\approx (d_2 \cdot s^2) \bmod q_l \end{aligned}$$

Algorithms in CKKS — Noise Analysis

	$\Delta \cdot x_1$
	$\Delta \cdot x_2$
	\vdots
	$\Delta \cdot x_{n/2}$

 \times

	$\Delta \cdot y_1$
	$\Delta \cdot y_2$
	\vdots
	$\Delta \cdot y_{n/2}$

 $=$

	$\Delta^2 \cdot x_1 y_1$
	$\Delta^2 \cdot x_2 y_2$
	\vdots
	$\Delta^2 \cdot x_{n/2} y_{n/2}$

CKKS Algorithm in Detail — Rescaling

在全同态加密中，密文的模 q 的大小决定了能够执行多少次同态乘法。每次同态乘法都会“消耗”模 q 的值，从而形成一系列逐步减小的模： q_L, q_{L-1}, \dots, q_1 （通常称之为模链）。

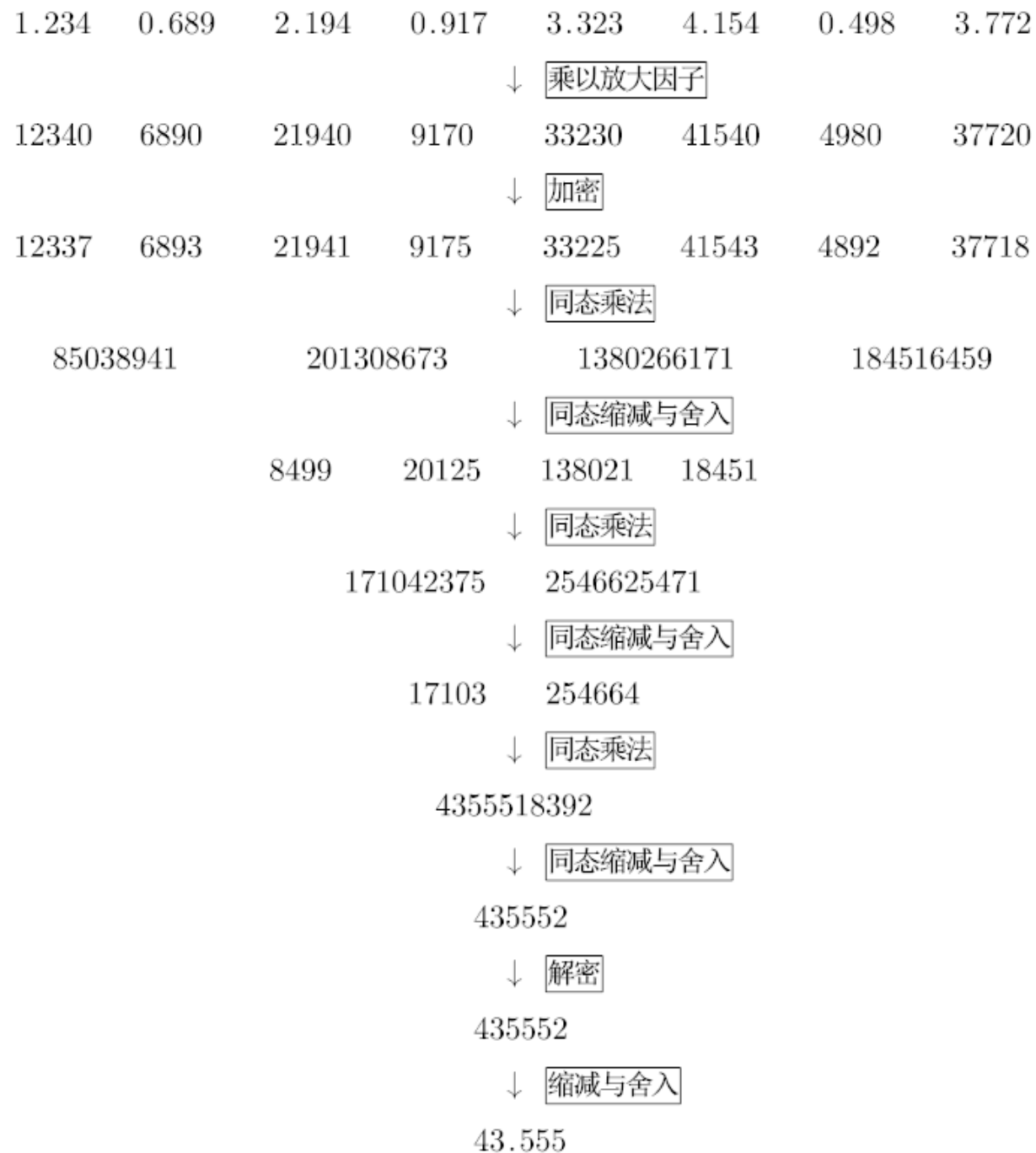
在 CKKS 算法中，该如何设置模 q 的值呢？假设需要执行 L 次乘法，放大因子为 Δ ，注意 Δ 需要保持小数部分位数的精度。那么可以设置 $q_L = q_0 \cdot p^L = q_0 \cdot \Delta^L$ ，其中 $q_0 \geq \Delta$ ，这样就形成了一个模链， $q_i/q_{i-1} = \Delta$ 。第一次加密的密文模数是 Q ，之后每执行一次同态乘法， Δ 都会增加，这里就可以对模数除以一个 p 来抵消掉这个增加的 Δ ，从而降低深度。例如，我们需要小数部分保留 30 位的精度，整数部分保留 10 位的精度，执行 8 次同态乘法，那么可以设置 $\Delta = 2^{30}$ ， $q_0 = 2^{30+10}$ ，则 $q = 2^{30 \times 8 + 40} = 2^{280}$ 。

具体的再缩减技术也很简单，就是密文除以放大因子 Δ ，然后再执行舍入操作。例如，密文 \mathbf{c} 对应的模为 q_i ，执行相应的再缩减操作



$$RS_{i \rightarrow i-1}(\mathbf{c}) = \left\lfloor \frac{q_{i-1}}{q_i} \mathbf{c} \right\rfloor \bmod q_{i-1}$$

同态乘法后需要先执行再线性化操作，然后执行再缩减操作，过程如下：

（1）执行同态乘法 → （2）执行再线性化操作 → （3）执行再缩减操作



CKKS 近似同态思想示例



感谢聆听

The End

