第3章 同态加密

学习要求：掌握同态加密的特点、定义和分类；了解同态加密的发展历史；了解典型方案的构造思想，理解同态加密的应用场景，能够运用不同类型的同态加密解决实际问题；理解自举的概念；掌握理想格的概念及格上的两类难题；了解 BGN，Gentry 和 CKKS 方案设计的主要思想；掌握基于 Paillier 的隐私信息获取的应用示例，以及基于 SEAL 的 CKKS 的开发案例。

课时：2 课时

建议授课进度：3.1 节～3.2 节用 1 课时，3.3 节～3.5 节用 1 课时

3.1 基本概念

3.1.1 定义

同态加密是一种加密算法，它可以通过对密文进行运算得到加密结果，解密后与明文运算的结果一致，如图 3-1 所示。

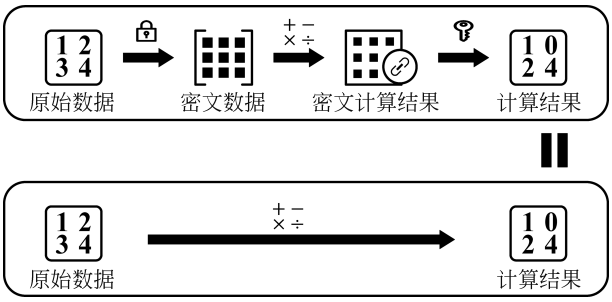

图 3-1 同态加密效果图

同态加密主要基于公钥密码体制构建，它允许将加密后的密文发给任意的第三方进行计算，并且在计算前不需要解密，可以在不需要密钥方参与的情况下，在密文上直接进行计算。

同态加密方案由 KeyGen，Encrypt，Decrypt 和 Evaluate 4 个函数构成。

(1) KeyGen(λ)→(pk，sk)：密钥生成函数；在给定加密参数 λ 后，生成公钥/私钥对(pk，sk)。

(2) Encrypt(pk,pt)→ct：加密函数；使用给定公钥 pk 将目标明文数据 pt 加密为密文 ct。

(3) Decrypt(sk,ct)→pt：解密函数；使用给定密钥 sk 将目标密文数据 ct 解密为明文 pt。

(4) Evaluate(pk,Π,ct_1,ct_2,…)→(ct'_1,ct'_2,…)：求值函数；给定公钥 pk 与准备在密文上进行的运算函数 Π，求值函数将一系列的密文输入(ct_1,ct_2,…)转换为密文输出(ct'_1,ct'_2,…)。

求值函数是同态加密方案不同于传统加密方案的部分。它的参数 Π 支持的运算函数种类决定了该同态加密方案支持的同态运算操作。

在给定以上 4 个函数后，同态加密方案应满足正确性、语义安全性和简短性。

(1) 正确性：一个同态加密系统必须要是正确的。具体来说，也就是加密之后的密文可以被成功解密，并且求值函数输出的密文也可以成功解密回原文。

(2) 语义安全性：同态加密系统输出的密文必须难以分辨。具体来说，如果有一个网络窃听者看到了所有的密文，那么这个窃听者并不能分辨出哪个密文是对应哪个原文的。

(3) 简短性：同态加密的求值函数输出的密文的长度需要在一个可以控制的长度范围内，确保了同态加密系统的实用性。

3.1.2　分类

根据同态加密算法所支持的同态操作种类和次数，可以将现有同态加密方案分为以下几种类型。

(1) 半同态加密(partial homomorphic encryption，PHE)：仅支持单一类型的密文域同态运算(加或乘同态)。

(2) 类同态加密(somewhat homomorphic encryption，SHE)：能够支持密文域有限次数的加法和乘法同态运算。

(3) 层级同态加密(leveled homomorphic encryption，LHE)：能同时支持多种同态操作(加或乘同态)，并可以在安全参数中定义能够执行的操作次数上限。一般允许的操作次数越大，该同态加密方案的密文空间开销及各类操作的时间复杂度就越大。

(4) 全同态加密(fully homomorphic encryption，FHE)：能够实现任意次密文的加、乘同态运算。

3.1.3　发展历史

同态加密的发展历史如表 3-1 所示。

表 3-1　同态加密的发展历史

类　型	算法	时间	说　明
半同态加密	RSA 算法	1977 年	非随机化加密，具有乘法同态性的原始算法面临选择明文攻击
	ElGamal 算法	1985 年	随机化加密，乘法同态
	Paillier 算法	1999 年	加法同态，在联邦学习中广泛应用

续表

类　型		算法	时间	说　明
类同态加密		BGN 方案	2005 年	支持任意次加法和一次乘法操作的同态运算
全同态加密	第一代	Gentry 方案	2009 年	自举操作，性能差
	第二代	BGV 方案	2012 年	基于算术电路，基于模归约提升了自举性能
		BFV 方案	2012 年	基于算术电路，使用 SIMD 操作提升了自举性能
	第三代	GSW 方案	2013 年	支持任意布尔电路，基于近似特征向量
		FHEW 方案	2015 年	支持任意布尔电路，可实现快速比较
		TFHE 方案	2016 年	支持任意布尔电路，基于近似特征向量
	第四代	CKKS 方案	2017 年	可实现浮点数近似计算

1. 半同态加密

(1) **乘法同态加密**是指存在有效算法$\otimes$，使得 $\mathrm{Enc}(x)\otimes\mathrm{Enc}(y)=\mathrm{Enc}(xy)$或者 $\mathrm{Dec}(\mathrm{Enc}(x)\otimes\mathrm{Enc}(y))=xy$ 成立，并且不泄露 x 和 y。

典型乘法同态加密算法是 RSA 算法和 ElGamal 算法。以 RSA 算法为例，如果 $c_1=m_1^e \bmod n$，$c_2=m_2^e \bmod n$，那么 $c_1c_2=m_1^e m_2^e \bmod n=(m_1m_2)^e \bmod n\equiv\mathrm{Enc}(m_1m_2)$。

(2) **加法同态加密**是指存在有效算法$\oplus$，使得 $\mathrm{Enc}(x)\oplus\mathrm{Enc}(y)=\mathrm{Enc}(x+y)$或者 $\mathrm{Dec}(\mathrm{Enc}(x)\oplus\mathrm{Enc}(y))=x+y$ 成立，并且不泄露 x 和 y。

典型加法同态加密算法是 Paillier 算法，详见 3.2 节描述。

注意：加法和乘法同态是相对明文而言所执行的操作，而非密文上执行的运算形式。

2. 类同态加密

类同态加密方案能够同时支持加法和乘法的同态操作。但由于它生成的密文随着操作次数的增加而逐渐增大，能够在密文上执行的同态操作次数是有上限的。

典型的类同态加密方案是 Boneh、Goh 和 Nissim 在 2005 年提出的 Boneh-Goh-Nissim (BGN)方案[①]，它支持在密文大小不变的情况下进行任意次数的加法和一次乘法。该方案中的加法同态基于类似 Paillier 算法的思想，而一次乘法同态基于双线性映射的运算性质。虽然该方案是双同态的(同时支持加法同态和乘法同态)，但只能进行一次乘法操作。

3. Gentry 方案(第一代全同态加密方案)

在同态加密概念提出后的 30 年间，并没有真正能够支持无限制的各类同态操作的全同态加密方案问世。

2009 年，Gentry[②] 基于所提出的类同态加密方案，提出了自举(bootstrapping)技术，可以将满足条件的类同态加密方案改造成全同态加密方案。其基本思想是在类同态加密算法的基础上引入自举方法来控制运算过程中的噪声增长(类同态加密算法操作次数过多会导

① BONEH D, GOH E J, NISSIM K. Evaluating 2-DNF formulas on ciphertexts[C]//Theory of Cryptography Conference, 2005: 325-341.

② GENTRY C. A fully homomorphic encryption scheme[M]. Stanford University, 2009.

致噪声过大而无法解密),这也是第一代全同态加密方案的主流模型。

为了避免多次运算使得噪声扩大,Gentry 方案采用了计算一次就消除一次噪声的方法,而消除噪声的方法还是使用的同态运算。但是,由于解密过程本身的运算十分复杂,运算过程中也会产生大量噪声,因此,Gentry 方案性能极差,一次同态乘法可能需要 30min。

现在的第四代全同态加密解决方案要比 Gentry 提出的方案要好得多,性能大概提高了 100 万倍,并且已经开始制定相关的标准。

4. BGV 和 BFV 方案(第二代全同态加密方案)

第二代全同态加密方案主要包括 BGV[①] 和 BFV[②],通常基于容错学习问题(learning with error,LWE)和环上容错学习问题(ring learning with error,RLWE)假设,其安全性基于格困难问题。

第二代方案主要是解决自举操作带来的昂贵操作,通过引入层级同态加密等来提升性能。此外,第二代全同态加密还提出了单指令多数据(single instruction multiple data,SIMD)操作,通过批量处理来提高吞吐量,极大降低了均摊复杂度。简单来说,SIMD 操作把密文切出上千个槽,把上千个明文放在这些密文槽中,这样,就可以并行处理各个槽中的数据了。在此基础上,还可以利用同构性置换各个槽中的数据,各个槽中的数据也可以相互运算。

第二代全同态加密方案的性能已经提升了很多,每个明文位的自举时间约为 0.9ms,自举一个密文能在 10s 左右完成,具有了一定的实用性。HElib 和 SEAL(simple encrypted arithmetic library)两个全同态加密开源库均支持 BGV 和 BFV 方案。

5. TFHE 等方案(第三代全同态加密方案)

GSW[③],FHEW[④] 和 TFHE[⑤] 是第三代同态加密方案重要的代表作。与第二代 FHE 方案相比,自举的性能得到大幅度提升,在常见的台式机平台上速度可以达到毫秒级别;但同时因为缺少第二代 FHE 的 SIMD 特性,FHEW 只能处理若干位(典型值为 2～7)的加法和乘法操作,也就是说同态乘法的性能较差。

6. CKKS 等方案(第四代全同态加密方案)

CKKS 方案[⑥]支持针对实数或复数的浮点数加法和乘法同态运算,但是得到的计算结果是近似值。因此,它适用于不需要精确结果的场景。支持浮点数运算这一功能在实际中

---

① BRAKERSKI Z, GENTRY C, VAIKUNTANATHAN V. (Leveled) fully homomorphic encryption without bootstrapping[J]. ACM Transactions on Computation Theory, 2014,6(3): 1-36.

② BRAKERSKI Z. Fully homomorphic encryption without modulus switching from classical GapSVP[C]//Annual cryptology conference, 2012: 868-886.

③ GENTRY C, SAHAI A, WATERS B. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based[C]//Annual cryptology conference, 2013: 75-92.

④ DUCAS L, MICCIANCIO D. FHEW: Bootstrapping homomorphic encryption in less than a second[C]//Annual international conference on the theory and applications of cryptographic techniques, 2015: 617-640.

⑤ CHILLOTTI I, GAMA N, GEORGIEVA M, et al. TFHE: Fast fully homomorphic encryption over the torus[J]. Journal of Cryptology, 2020, 33(1): 1-58.

⑥ CHEON J H, KIM A, KIM M, et al. Homomorphic encryption for arithmetic of approximate numbers[C]//Advances in Cryptology-ASIACRYPT 2017, 2017: 409-437.

有非常重要的作用，如实现机器学习模型训练等。这个方案的性能也非常优异，大多数算法库都实现了 CKKS。

注意：有关全同态加密的发展历程，可以关注 Gentry 在 EUROCRYPT 2021 上的邀请报告，网络上有中文翻译版。

3.2 半同态 Paillier 方案

Paillier 加密算法[①]是 Paillier 等于 1999 年提出的一种基于判定 n 阶剩余类难题的典型密码学加密算法，具有加法同态性，是半同态加密方案。

3.2.1 数学基础

1. 卡迈克尔函数

在数论中，卡迈克尔函数的定义如下：设 $\gcd(a,n)=1$，gcd 为求最大公约数，使得 $a^m \equiv 1 \bmod n$ 成立的最小正整数 m，将 m 记作 $\lambda(n)$。对于 $n=pq$，p 和 q 都是素数，则有 $\lambda(n)=\mathrm{lcm}(p-1,q-1)$，lcm 为求最小公倍数。

在数论中，对正整数 n，欧拉函数是小于 n 的正整数中与 n 互质的数的数目。显然 $\phi(1)=1$，而对于 $m>1$，$\phi(m)$ 就是 $\{1,2,\cdots,m-1\}$ 中与 m 互素的数的个数，如果 p 是素数，则有 $\phi(p)=p-1$。对于 $n=pq$，p 和 q 都是素数，则有 $\phi(n)=(p-1)(q-1)$。显然，如果 $p-1$ 和 $q-1$ 也分别为素数的话，那么 $\phi(n)=\lambda(n)$，否则，$\phi(n)$ 是 $\lambda(n)$ 的倍数。

表 3-2 是卡迈克尔函数 $\lambda(n)$ 与欧拉函数 $\phi(n)$ 的对比表。

表 3-2 卡迈克尔函数 $\lambda(n)$ 与欧拉函数 $\phi(n)$ 的对比表

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\lambda(n)$	1	1	2	2	4	2	6	2	6	4	10	2	12	6	4	4
$\phi(n)$	1	1	2	2	4	2	6	4	6	4	10	4	12	6	8	8

1）示例

8 的卡迈克尔函数是 2，即 $\lambda(8)=2$，即对于任意的 a 满足 $\gcd(a,8)=1$，有 $a^m \equiv 1 \bmod 8$，也就是说 $1^2 \equiv 1 \bmod 8$，$3^2 \equiv 1 \bmod 8$，$5^2 \equiv 1 \bmod 8$，$7^2 \equiv 1 \bmod 8$。

而对于欧拉函数来说，$\phi(8)=4$，因为欧拉函数是计算与 8 互素的数的数量，即 1，3，5，7。

对于 $n=15$，因为 $n=3\times5$，令 $p=3$，$q=5$，$\lambda(15)=\mathrm{lcm}(2,4)=4$，$\phi(15)=2\times4=8$。

2）卡迈克尔函数的性质

设 $n=pq$，其中：p 和 q 是大素数。那么 $\phi(n)=(p-1)(q-1)$，$\lambda(n)=\mathrm{lcm}(p-1,q-1)$。为便于描述，用 λ 表示 $\lambda(n)$。

对于任意 $g \in \mathbf{Z}_{n^2}^*$，有如下性质：

① PAILLIER P. Public-key cryptosystems based on composite degree residuosity classes[C]//International conference on the theory and applications of cryptographic techniques, 1999: 223-238.

$$\begin{cases} g^{\lambda} \equiv 1 \bmod n \\ g^{n\lambda} \equiv 1 \bmod n^2 \end{cases}$$

具体推导如下。

根据 λ 的定义，可得 $\lambda = k_1(p-1) = k_2(q-1)$

根据费马小定理 $g^{p-1} = 1 \bmod p$，可得

$$g^{\lambda} = g^{k_1(p-1)} = (g^{p-1})^{k_1} \equiv 1 \bmod p$$

同理 $g^{\lambda} \equiv 1 \bmod q$

所以 $g^{\lambda} \equiv 1 \bmod pq = 1 \bmod n$

所以 $g^{\lambda} = 1 + kn$，$k \in \mathbf{Z}_n^*$

结合上式及二项式定理，可得

$$g^{n\lambda} \bmod n^2 = (1+kn)^n \bmod n^2 \equiv (1+kn^2) \bmod n^2 \equiv 1 \bmod n^2$$

推导中使用了如下性质：对于 $1+n \in \mathbf{Z}_{n^2}^*$，有

$$(1+n)^2 \equiv 1 + 2n + n^2 \equiv (1+2n) \bmod n^2$$

$$(1+n)^3 \equiv 1 + 3n + n^3 \equiv (1+3n) \bmod n^2$$

$$(1+n)^v \equiv 1 + vn + \cdots \equiv (1+vn) \bmod n^2$$

2. 判定复合剩余假设

剩余类：也称同余类，指全体整数按照对一个正整数的同余关系而分成的类。对于一个整数 m，可以把所有整数分成 m 类，每类模 m 后余数都相同，每一类都叫作 m 的一个剩余类。比如，给定整数 5，有 5 个剩余类，对 0 同余的有$\{-5, 0, 5, \cdots\}$。

复合剩余类：如果存在一个数 $x \in \mathbf{Z}_{n^2}^*$，那么符合公式 $z = x^n \bmod n^2$ 的数 z，称为 x 模 n^2 的 n 阶剩余类。或者说，如果数 z 被称为 x 的模 n^2 的 n 阶剩余类，则存在一个数 $x \in \mathbf{Z}_{n^2}^*$，使得 $z = x^n \bmod n^2$。

判定复合剩余假设(decisional composite residuosity assumption，DCRA)：设 $n = pq$，p 与 q 为两个大素数，对于任意给定的整数 z，判断它是不是模 n^2 的 n 阶剩余类是一个难解问题。

3.2.2　方案构造

1. 算法描述

1) 密钥生成

(1) 随机选择两个素数 p 和 q，尽可能地保证 p 和 q 的长度接近或相等(安全性高)。

(2) 计算 $n = pq$ 和 $\lambda = \mathrm{lcm}(p-1, q-1)$，其中 lcm 表示最小公倍数。

(3) 随机选择 $g \in \mathbf{Z}_{n^2}^*$，考虑计算性能优化，通常会选择 $g = n+1$。

(4) 计算 $\mu = [L(g^{\lambda} \bmod n^2)]^{-1} \bmod n$，其中 $L(x) = \dfrac{x-1}{n}$。

(5) 公钥为(n, g)。

(6) 私钥为(λ, μ)。

2) 加密算法

对于任意明文消息 $m \in \mathbf{Z}_n$，任意选择一个随机数 $r \in \mathbf{Z}_n^*$，计算得到密文

$$c=E(m)=g^m r^n \bmod n^2$$

注意：密文 c 要比明文 m 更长。

3）解密算法

对于密文 $c\in \mathbf{Z}_{n^2}^*$，计算得到明文

$$m=D(c)=L(c^\lambda \bmod n^2)\mu \bmod n$$

2. 正确性

依据卡迈克尔函数的性质，对于任意 $g\in \mathbf{Z}_{n^2}^*$，$n=pq$ 和 $\lambda=\mathrm{lcm}(p-1,q-1)$，有

$$\begin{cases} g^\lambda \equiv 1 \bmod n \\ g^{n\lambda} \equiv 1 \bmod n^2 \end{cases}$$

如 3.2.1 节所述，$g^\lambda=1+kn$，$k\in \mathbf{Z}_n^*$，基于上述三个性质，解密过程推导如下。

$$\begin{aligned} D(c) &= L(c^\lambda \bmod n^2)\mu \bmod n \\ &= L((g^m r^n)^\lambda \bmod n^2)\mu \bmod n \end{aligned}$$

参考 $r^{n\lambda}\equiv 1 \bmod n^2$ 性质可得

$$\begin{aligned} D(c) &= L((g^\lambda)^m \bmod n^2)(L(g^\lambda \bmod n^2))^{-1} \bmod n \\ &= L((1+kn)^m \bmod n^2)\,(L(1+kn)\bmod n^2)^{-1} \bmod n \end{aligned}$$

参考 $(1+n)^v \equiv 1+vn+\cdots \equiv (1+vn) \bmod n^2$ 可得

$$\begin{aligned} D(c) &= L((1+mkn)\bmod n^2)\,(L(1+kn)\bmod n^2)^{-1} \bmod n \\ &= mkk^{-1} \bmod n \\ &= m \end{aligned}$$

3. 加法同态性

对于任意明文 $m_1, m_2\in \mathbf{Z}_n$ 和任意 $r_1, r_2\in \mathbf{Z}_n^*$，对应密文 $c_1=E(m_1)$，$c_2=E(m_2)$，满足

$$c_1 c_2 = g^{m_1} r_1^n g^{m_2} r_2^n \bmod n^2 = g^{m_1+m_2}(r_1 r_2)^n \bmod n^2$$

解密后得到

$$D(c_1 c_2)=D(g^{m_1+m_2}(r_1 r_2)^n \bmod n^2)=m_1+m_2$$

即 $c_1 c_2=m_1+m_2$，也就是，**密文乘等于明文加**。

注意：这里定义的密文加法运算形式是乘法运算，但是因为运算的结果是明文相加，因此是加法同态。加法或乘法同态是相对明文而言所执行的操作，而非密文上执行的运算形式。

4. 标量乘同态性

对于明文 $m_1\in \mathbf{Z}_n$ 及其密文 c_1，给定一个整数 $a\in \mathbf{Z}_n$，满足

$$D(c_1^a \bmod n^2)=D(g^{m_1 a}(r^a)^n \bmod n^2)=m_1 a$$

注意：这里定义的密文标量乘运算形式是指数运算 c_1^a，但是因为运算的结果解密是常数乘明文 $m_1 a$，因此是标量乘法。

3.2.3 应用示例

1. 典型应用

半同态加密虽然还不能同时支持加法和乘法运算，不能支持任意地计算，但是因为其与

全同态相比，具有较高性能，因此，仍然具有极为广泛的应用场景，且在现实应用中起到了重要的作用。一类典型的应用体现在隐私保护的数据聚合上。由于加法同态加密可以在密文上直接执行加和操作，不泄露明文，在多方协作的统计场景中，可完成安全的统计求和的功能。

1）联邦学习

在联邦学习（federated learning，FL）中，不同参与方训练出的模型参数可由一个第三方进行统一聚合。使用加法 PHE，可以在明文数据不出域且不泄露参数的情况下，完成对模型参数的更新，此方法已在实际中应用（如 FATE），如图 3-2 所示。

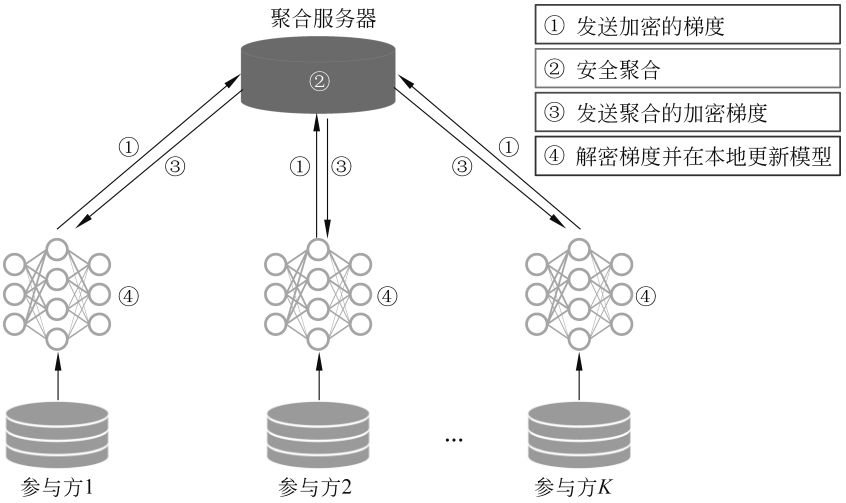

图 3-2　联邦学习中的安全聚合示例

2）隐私集合求和

在线广告投放的场景中，广告主（如商家）在广告平台（如媒体）投放在线广告，并希望计算广告点击的转化收益。然而，广告点击数据集和购买数据集分散在广告主和广告平台两方。使用加法 PHE 结合隐私集合求和（private intersection-sum-with-cardinality，PIS-C）协议可以在保护双方隐私数据前提下，计算出广告的转化率。如图 3-3 所示，协议中的隐私保护求和功能依赖于广告主将自己的交易数据用 PHE 加密发送给广告平台，使得广告平台在看不到原始数据的前提下，完成对交集中数据金额的聚合。该方案已被 Google 落地应用。

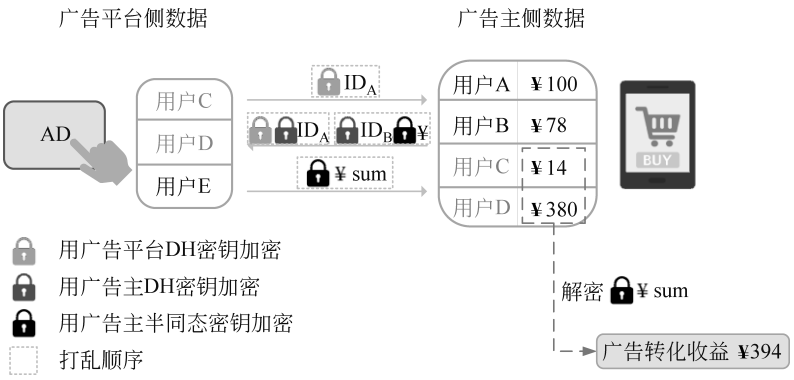

图 3-3　加法 PHE 在 PIS-C 中的应用

3）数据库统计查询

在加密数据库 SQL 查询场景，在数据库不可信的情况下，可以通过部署协议和代理来保护请求者的查询隐私。其中，PHE 可以用来完成安全数据求和、均值的查询。

除了上述场景，加法 PHE 还可被用于多种行为数据和效益数据分离的商业场景，在应用上有着很大的想象空间。

2. 实验环境安装

1）安装 Python 环境

在 Windows 操作系统下安装 Python 开发环境，可以进入官方网站 https://www.python.org/downloads/，下载 Windows 操作系统的 Python 安装包，下载后运行下载文件并按照安装向导的指示安装即可。

注意：安装时一定要勾选 Add python.exe to PATH 复选框，这样会使得安装后的 Python 程序路径直接加入时系统的环境变量中，在控制台可以直接使用 Python 命令。如果忘记勾选，则需要右击“我的电脑”图标，在弹出的快捷菜单中选择“属性”→“高级系统设置”→“环境变量”命令，在 Path 中将安装的路径手动输入。

安装完毕，打开控制台，输入 Python 命令，如图 3-4 所示。

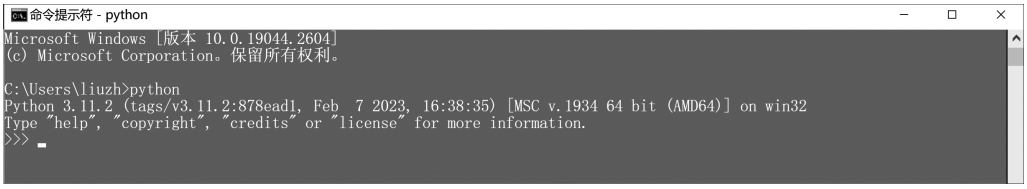

图 3-4　进入 Python 环境

这代表已经安装成功，并且进入 Python 运行环境。

（1）输入 Python 程序。

```
from phe import paillier
```

该命令将导入 phe 库的 paillier 功能，第一次执行会提示 ModuleNotFoundError：No module named 'phe'。这是因为，默认安装 Python 程序后，并没有安装 phe 库。

（2）输入退出命令。

```
exit()
```

该命令可以退出当前 Python 环境，切回控制台模式，如图 3-5 所示。

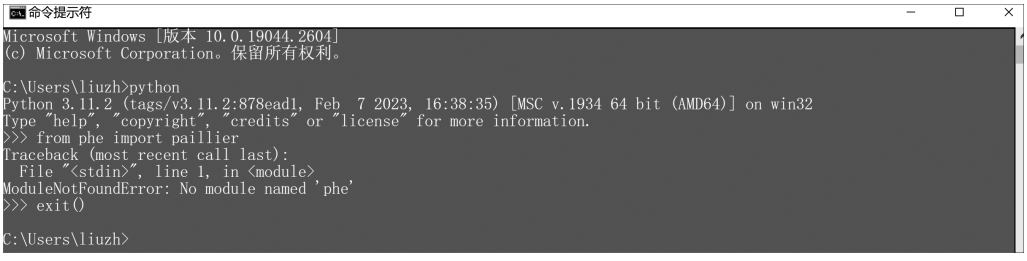

图 3-5　退出 Python 环境

2）安装 phe 库

输入如下命令。

```
pip install phe
```

完成 phe 库的安装，如图 3-6 所示。

```
C:\Users\liuzh>pip install phe
Collecting phe
  Using cached phe-1.5.0-py2.py3-none-any.whl (53 kB)
Installing collected packages: phe
Successfully installed phe-1.5.0

[notice] A new release of pip available: 22.3.1 -> 23.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

图 3-6　安装 phe 库

pip 是 Python 语言的一个安装库的工具，可执行文件在 Python 程序安装目录下可以找到。

3）验证环境正确性

再次进入 Python 环境，输入如下 Python 代码。

```
from phe import paillier
```

结果如图 3-7 所示。

```
C:\Users\liuzh>python
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from phe import paillier
>>>
```

图 3-7　验证环境正确性

如果不出现错误信息，说明环境安装成功。

4）编写 Python 程序并运行

可以用三种方式调试和编写 Python 程序。

（1）在控制台运行 Python 命令，逐行编写 Python 程序并运行。

（2）用文本编辑器编写完整的程序并保存为 x.py 文件，通过控制台命令 python x.py 的方式完成整个程序的调用。

（3）通过自带的集成开发环境 IDLE 完成开发和调试运行。通过开始菜单，找到 IDLE 并打开，选择 File→New File 菜单项可以新建一个文件，编辑程序并保存后，选择 Run→Run Module 菜单项运行，会看到运行的结果。

3. 简单示例

phe 库的使用说明详见 https://python-paillier.readthedocs.io/en/develop/usage.html#usage。

实验 3-1　基于 Python 语言的 phe 库完成加法和标量乘法的验证。

演示代码如下。

```
1.  from phe import paillier #开源库
2.  import time #做性能测试
```

```
3.
4.  ######################设置参数
5.  print("默认私钥大小:", paillier.DEFAULT_KEYSIZE)
6.  #生成公私钥
7.  public_key, private_key = paillier.generate_paillier_keypair()
8.  #测试需要加密的数据
9.  message_list = [3.1415926,100,-4.6e-12]
10.
11. ######################加密操作
12. time_start_enc = time.time()
13. encrypted_message_list = [public_key.encrypt(m) for m in message_list]
14. time_end_enc = time.time()
15. print("加密耗时 s:",time_end_enc-time_start_enc)
16. print("加密数据(3.1415926):",encrypted_message_list[0].ciphertext())
17.
18. ######################解密操作
19. time_start_dec = time.time()
20. decrypted_message_list = [private_key.decrypt(c) for c in encrypted_
    message_list]
21. time_end_dec = time.time()
22. print("解密耗时 s:",time_end_dec-time_start_dec)
23. print("原始数据(3.1415926):",decrypted_message_list[0])
24.
25. ######################测试加法和乘法同态
26. a,b,c = encrypted_message_list #a,b,c 分别为对应密文
27. a_sum = a + 5 #密文加明文,已经重载了+运算符
28. a_sub = a - 3 #密文加明文的相反数,已经重载了-运算符
29. b_mul = b * 6 #密文乘明文,数乘
30. c_div = c / -10.0 #密文乘明文的倒数
31. print("a+5 密文:",a.ciphertext()) #密文纯文本形式
32. print("a+5=",private_key.decrypt(a_sum))
33. print("a-3",private_key.decrypt(a_sub))
34. print("b*6=",private_key.decrypt(b_mul))
35. print("c/-10.0=",private_key.decrypt(c_div))
36. ##密文加密文
37. print((private_key.decrypt(a)+private_key.decrypt(b))==private_key.
    decrypt(a+b))
38. #报错,不支持 a*b,即两个密文直接相乘
39. #print((private_key.decrypt(a)+private_key.decrypt(b))==private_key.
    decrypt(a*b))
```

如上述代码所示：第一，Python 程序对运算符进行了承载，已经支持直接密文上的运算；第二，只支持明文的加法，不支持明文的乘法，最后一句如果将注释符去掉，将报错。

4. 隐私信息获取示例

实验 3-2 基于 Python 语言的 phe 库完成隐私信息获取的功能：服务器拥有多个数值，要求客户端能基于 Paillier 实现从服务器读取一个指定的数值并正确解密，但服务器不知道所读取的是哪一个。

首先，基于 Paillier 协议进行设计。

对 Paillier 的标量乘的性质进行扩展，可以知道：数值 0 的密文与任意数值的标量乘也是 0，数值 1 的密文与任意数值的标量乘将是数值本身。

基于这个特性，可以进行如下巧妙设计。

服务器：产生数据列表 message_list＝{m_1，m_2，…，m_n}。

客户端：

(1) 设置要选择的数据位置为 pos。

(2) 生成选择向量 select_list＝{0，…，1，…，0}，其中：仅有 pos 的位置为 1。

(3) 生成密文向量 enc_list＝{$E(0)$，…，$E(1)$，…，$E(0)$}。

(4) 发送密文向量 enc_list 给服务器。

服务器：

(1) 将数据与对应的向量相乘后累加得到密文

$$c = m_1 * \text{enc\_list}[1] + \cdots + m_n * \text{enc\_list}[n]$$

(2) 返回密文 c 给客户端。

客户端：解密密文 c 得到想要的结果。

然后，开发具体代码如下。

```
1.  from phe import paillier                    #开源库
2.  import random                               #选择随机数
3.
4.  ######################设置参数
5.  #服务器保存的数值
6.  message_list = [100,200,300,400,500,600,700,800,900,1000]
7.  length = len(message_list)
8.  #客户端生成公私钥
9.  public_key, private_key = paillier.generate_paillier_keypair()
10. #客户端随机选择一个要读的位置
11. pos = random.randint(0, length-1)
12. print("要读起的数值位置为:", pos)
13.
14. ######################客户端生成密文选择向量
15. select_list=[]
16. enc_list=[]
17. for i in range(length):
18.     select_list.append(i == pos)
19.     enc_list.append(public_key.encrypt(select_list[i]))
20. ######################服务器进行运算
21. c=0
22. for i in range(length):
23.     c = c + message_list[i] * enc_list[i]
24. print("产生密文:", c.ciphertext())
25.
26. ######################客户端进行解密
27. m=private_key.decrypt(c)
28. print("得到数值:", m)
```

扩展思考：在客户端保存对称密钥 k，在服务器存储 m 个用对称密钥 k 加密的密文，通过隐私信息获取方法得到指定密文后能解密得到对应的明文，如何设计实现？

3.3 类同态 BGN 方案

1994 年，Fellows 和 Koblitz 提出第一个同时支持同态加法和同态乘法的类同态加密方案[①]，其密文大小随着同态操作的次数呈指数级增长，且同态乘法的计算开销很大，无法投入实际使用。2005 年，BGN 加密方案[②]提出，它支持在密文大小不变的情况下进行任意次数的加法和一次乘法。

3.3.1 数学基础

1. 群

群是一种由元素的集合和一个二元运算组成的基本代数结构。若元素集合 G 和二元运算“$\cdot$”满足封闭性、结合律、单位元和逆元素四个要素，则称为群。

(1) 封闭性：对于所有集合 G 中的元素 a 和 b，$a \cdot b$ 的结果也在集合 G 中。

(2) 结合律：$(a \cdot b) \cdot c = a \cdot (b \cdot c)$ 对任意 $a, b, c \in G$ 成立。

(3) 单位元：集合 G 存在元素 e，满足 $e \cdot a = a \cdot e = a$，$\forall a \in G$。

(4) 逆元素：对于集合 G 中的任意一个元素 a，存在集合 G 中的另一个元素 b 使 $a \cdot b = b \cdot a = e$。

若一个群还满足交换律(即 $a \cdot b = b \cdot a$，对任意 $a, b \in G$ 成立)，则可进一步称为交换群或阿贝尔群。定义一个有限群的阶为群中元素的个数。对于二元运算“$\cdot$”，定义元素的乘方 a^2 为 $a \cdot a$，并以此推演出元素的更高次方。

若一个群 G 的每一个元素都可以被表达成群 G 中某一个元素 g 的次方 g^m，则称 G 为循环群，记作 $G = (g) = \{g^m \mid m \in \mathbf{Z}\}$，$g$ 被称为 G 的一个生成元，因为可以通过对 g 的不断自我运算来获得群中的所有元素。

注意：符号 $<g>$ 与符号 (g) 相同，也经常被定义为由 g 生成的循环群。

在一个有限群中，如果对不是生成元的其他元素 a 进行这种次方运算，它最终会循环遍历一个群 G 的子集。可以证明，所有元素的这种遍历都会经过单位元 e。将满足 $a^n = e$ 的最小正整数 n 称为 a 元素的阶，生成元的阶和群的阶相等。

以整数模 6 加法群 $\mathbf{Z}_6 = \{0,1,2,3,4,5\}$ 为例，其群的阶为 6，单位元为 0。6 个元素的阶分别是 1,6,3,2,3,6。其中：1,5 为生成元。

以元素 5 为例，经过模加运算有

$5^1 = 5$

$5^2 = (5+5) \pmod 6 = 4$

---

① FELLOWS M, KOBLITZ N. Combinatorial cryptosystems galore! [J]. Contemporary Mathematics, 1994, 168: 51.

② BONEH D, GOH E J, NISSIM K. Evaluating 2-DNF formulas on ciphertexts[C]//Theory of Cryptography Conference, 2005: 325-341.

$5^3=(5+5+5)(\mathrm{mod}6)=3$

$5^4=(5+5+5+5)(\mathrm{mod}6)=2$

$5^5=(5+5+5+5+5)(\mathrm{mod}6)=1$

$5^6=(5+5+5+5+5+5)(\mathrm{mod}6)=0=e$

故称元素 5 的阶为 6，为群 G 的生成元。

可以很容易发现循环群的一个特性，即由生成元 g 构建的循环群很容易满足加法同态：$g^{r_1}\cdot g^{r_2}=g^{(r_1+r_2)}$。根据上述例子，很容易验证：对于明文 1 和 2，对应的密文是 5^5 和 5^4，因为这里的运算符“·”就是加法，显然，$5^5\cdot 5^4=5^9=(5^6\cdot 5^3)(\mathrm{mod}6)=5^3=5^{(1+2)}=3$。

2. 环

在群的基础上，还可以使用两种运算和元素集合 R 来构建环(ring)，这两种运算一般写作“+”和“·”。

(1) “+”一般表示环上的加法，其对应的单位元通常为 0，由其定义的群为加法群，群上的两个相同运算的“+”运算，可以记作 $a+a=2a$。

(2) “·”一般表示环上的乘法，其对应的单位元通常记作 e，由其定义的群为乘法群，群上的两个相同运算的“·”运算，可以记作 $a\cdot a=a^2$。

环可以认为是在加法交换群之上增加了乘法运算“·”，且满足如下性质。

(1) 封闭性：对于所有 R 中元素 a 和 b，$a\cdot b$ 的结果也在 R 中。

(2) 结合律：$(a\cdot b)\cdot c=a\cdot(b\cdot c)$对任意 $a,b,c\in R$ 成立。

(3) 单位元：R 存在元素 e，满足 $e\cdot a=a\cdot e=e$，$\forall a\in R$，该元素常被称为乘法单位元。

(4) 分配律：乘法操作可以在加法之间进行分配。即给出任意 $a,b,c\in R$，有 $a\cdot(b+c)=a\cdot b+a\cdot c$ 和$(b+c)\cdot a=b\cdot a+c\cdot a$。

事实上，满足上述封闭性和结合律的$(R,\cdot)$构成一个半群。再加上单位元，就构成一个幺半群。如果再有逆元素，就构成了一个群。

$(R,+,\cdot)$若满足：

(1) $(R,+)$构成交换群；

(2) $(R,\cdot)$ 构成幺半群；

(3) $(R,+,\cdot)$ 满足分配律。

则其构成一个环。

在一个环$(R,+,\cdot)$中，若其子集 I 与其加法构成子群$(I,+)$，且满足 $\forall i\in I,r\in R$，$i\cdot r\in I$，则称 I 为环 R 的一个右理想(right ideal)。若 $\forall i\in I,r\in R,r\cdot i\in I$ 则称 I 为环 R 的一个左理想(left ideal)。若同时满足左右理想，则称 I 为环 R 上的一个理想(ideal)。

理想对内具有乘法封闭性，对外具有乘法吸收性。

3. 域

域是环的一种特殊形式，它要求乘法运算必须满足交换律，因此域是交换环，还要求域中的除 0 外的所有元素都有乘法逆元素。

有限域，就是一个包含有限个元素的域。一个有限域的具体例子就是模 p 的整数域，其中：p 是一个素数。通常有限域可表示为 Z_p。

从群到环，再到域，是一个条件逐渐收敛的过程。

4. 双线性映射

一个双线性映射是由两个向量空间上的元素，生成第三个向量空间上一个元素的函数，并且该函数对每个参数都是线性的。若 $B: V\times W\rightarrow X$ 是一个双线性映射，则 V 固定，W 可变时，W 到 X 的映射是线性的；W 固定，V 可变时，V 到 X 的映射也是线性的。也就是说，保持双线性映射中的任意一个参数固定，另一个参数对 X 的映射都是线性的。

存在一个加法循环群 G_1 和乘法循环群 G_2，这两个群的阶都为大素数 q。定义 $e: G_1\times G_1\rightarrow G_2$ 为这两个循环点群之间的一个双线性映射，且该映射满足如下三个性质。

(1) 双线性：对于所有的 $P,Q\in G_1$ 和 $a,b\in \mathbf{Z}_q^*$，有

$$e(aP,bQ)=e(P,Q)^{ab}$$

$$e[(a+b)P,Q]=e(P,Q)^a\cdot e(P,Q)^b$$

其中，$\mathbf{Z}_q^*$ 表示不包含 0 的整数集，$\mathbf{Z}$ 表示整数集，q 表示阶，* 表示不包含 0 元素。

(2) 非退化性：e 为非平凡映射，即 e 不会将 $G_1\times G_1$ 的所有值映射到 G_2 的单位元。

(3) 可计算性：具有有效的算法对于任何的 $P,Q\in G_1$ 能够计算 $e(P,Q)$。

满足如上三个性质的双线性映射就称为可采纳的双线性映射。

Boneh 等给出了关于双线性映射更具体的描述，提出了与双线性相关的数学难题，并用于设计基于身份加密、基于属性的加密等密码原语。

5. 子群判定问题

子群判定问题是指给定 (n,G,G_1,e)，其中：群 G,G_1 具有相同的阶 $n=pq$；$e: G\times G\rightarrow G_1$ 是一个双线性映射，给定一个元素 $x\in G$，如果 x 的阶是 p，则输出 1，否则输出 0。

上述问题也可以描述为一个阶为 $n=pq$（p,q 为素数）的合数阶群里，判定一个元素是否属于某个阶为 p 的子群的问题。

该判定问题为困难问题，BGN 方案的实现就是基于子群判定问题。

3.3.2 方案构造

BGN 能够同时支持加法和乘法的关键原因在于，它提出了一套能够构建在两个群 G 和 G_1 之间的双线性映射 $e: G\times G\rightarrow G_1$ 的方法。BGN 提出的方法能够生成两个阶相等的乘法循环群 G 和 G_1，并建立其双线性映射关系 e，且满足当 g 是 G 的生成元时，$e(g,g)$ 为 G_1 的生成元。

在执行乘法之前，密文属于群 G 中的元素，可以利用群的二元操作进行密文的加法同态操作。密文的乘法同态操作通过该双线性映射函数，将密文从群 G 映射到 G_1 的元素当中。执行乘法同态操作之后，处于 G_1 的密文仍然能够继续使用同态加法。

1. 密钥生成

(1) 给出安全参数，选择大素数 q_1,q_2 并获得合数 $n=q_1q_2$，BGN 将构建两个阶为 n 的循环群 G,G_1 和双线性映射关系 $e: G\times G\rightarrow G_1$。

(2) 从 G 中随机选取两个生成元 g,u，并获得 $h=u^{q_2}$。可知 h 为某阶为 q_1 的 G 的子群的生成元。

(3) 公钥设置为(n, G, G_1, e, g, h)，私钥设置为q_1。

2. 加密

对于消息明文m(某小于q_2的自然数)，随机抽取 0 到n之间的一个整数r，生成密文

$$c = E(m) = g^m h^r \in G$$

3. 解密

使用私钥q_1，首先计算

$$c^{q_1} = (g^m h^r)^{q_1} = (g^m u^{q_2 r})^{q_1} = g^{mq_1} u^{q_2 q_1 r} = g^{mq_1} u^{nr} = (g^{q_1})^m$$

然后，计算离散对数得到明文$m = \log_{g^{q_1}}(c^{q_1})$。

3.3.3　同态性

1. 密文上的同态加法性质

由生成元g构建的循环群很容易构造加法同态，$g^{r_1} \cdot g^{r_2} = g^{(r_1 + r_2)}$。

对于两个密文$c_1 = g^{m_1} \cdot h^{r_1}$和$c_2 = g^{m_2} \cdot h^{r_2}$，很明显$c_1 \cdot c_2 = g^{(m_1 + m_2)} \cdot h^{(r_1 + r_2)}$。

2. 密文上的同态乘法

密文上的同态乘法，则通过双线性映射函数实现，$e(u^a, v^b) = e(u, v)^{ab}$。

在密钥生成的时候，定义$g_1 = e(g, g)$和$h_1 = e(g, h)$，且将h写作$h = g^{\alpha q_2}$(因为g可生成u：$u = g^{\alpha}$)，定义对c_1和c_2的同态乘法运算为

$$e(c_1, c_2) h_1^r = e(g^{m_1} h^{r_1}, g^{m_2} h^{r_2}) h_1^r = g_1^{m_1 m_2} h_1^{\hat{r}} \in G_1$$

其中：$\hat{r}$是前文提到的随机抽取的 0 到n之间的整数r。

由此可见，经过同态乘法之后的密文从G转移到了G_1，其解密过程在G_1上完成，即G_1的生成元$g_1 = e(g, g)$替代g。在群G_1上依然可以进行乘法同态操作，所以 BGN 支持同态乘法运算之后的同态加法运算。

但是，因为没有下一个群可以继续映射，BGN 加密的密文只能够支持一次同态乘法运算。

3.4　全同态典型方案

3.4.1　数学基础

1. 格的定义

给定一个n维向量空间$\mathbf{R}^n$，格(lattice)是其上的一个离散加法子群。

根据线性代数知识，可以构造一组n个线性无关的向量$\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \cdots, \boldsymbol{v}_n \in \mathbf{R}^n$。基于该组向量的整数倍的线性组合，可以生成一系列的离散点，即

$$L(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \cdots, \boldsymbol{v}_n) = \left\{ \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i \mid \alpha_i \in \mathbf{Z} \right\}$$

这些元素集合和对应的加法操作$(L, +)$称为格。这组线性无关的向量$\boldsymbol{B}$(即$\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \cdots, \boldsymbol{v}_n$)称为格基，其向量个数称为格的维度。

2. 格的示例

格基$(1,0)^T$与$(0,1)^T$可以产生二维空间的所有整数格，如图 3-8(a)所示。同时，使用格基$(1,0)^T$与$(1,1)^T$同样可以生成二维空间的所有整数格，如图 3-8(b)所示。也就是说，一个格的基向量可以有多个，图 3-8(a)的基向量正交程度好一些，称为“好基”，而图 3-8(b)的基向量正交程度坏一些，称为“坏基”。

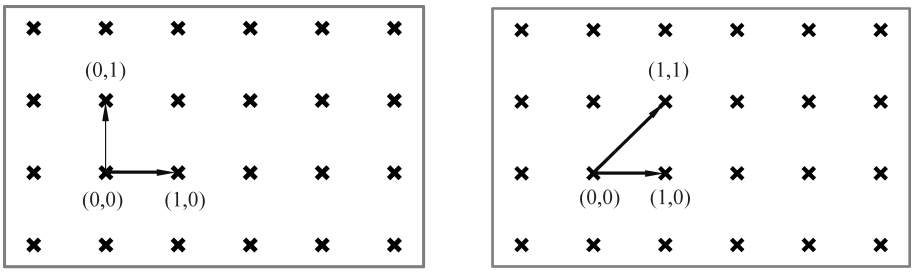

图 3-8　二维空间的格示例

再如，格基$(1,1)^T$与$(2,0)^T$不能产生二维空间的所有整数格。如图 3-9 所示，标有“×”号的为可产生的格，其横纵坐标相加为偶数。

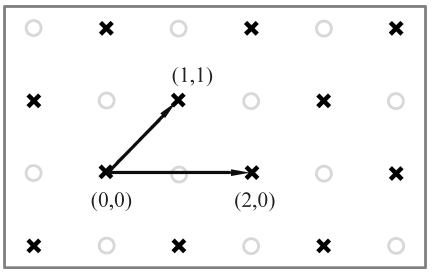

图 3-9　二维空间的格示例

上面都是简单的二维例子，一个格可以由无限多的维度和无限多的向量组成，所以虽然二维看起来非常简单，但是随着基向量和维度的数量增加时，问题很快会变得非常复杂。一般来说，对于达到足够安全性的方案，格的维度在 1000 左右。

注意：通过图 3-9 可以看出，并非所有的基都能生成一个格上的所有元素，而且通过“坏基”推测一个“好基”是一个难题。

3. 格上的难题

尽管格也由基扩展获得，它和向量空间最大的不同在于，**它的系数限制为整数**，从而生成一系列离散的空间向量。格上的向量的离散性质催生了一系列新的难题。

格上的主要难题是最近向量问题(closest vector problem，CVP)和最短向量问题(shortest vector problem，SVP)。

定义 3-1(最近向量问题)　给定一个格 L 和一个在向量空间 $\mathbf{R}^n$ 中但不在格 L 中的向量 $\boldsymbol{w}\in\mathbf{R}^n$，试图找到一个离 $\boldsymbol{w}$ 最近的向量 $\boldsymbol{v}\in L$，即与 $\boldsymbol{w}$ 的欧氏距离最小的向量。

欧氏距离也称欧几里得距离，以古希腊数学家欧几里得命名，是最常见的距离度量，衡量的是多维空间中两个点之间的绝对距离。例如，在二维和三维空间中的欧氏距离就是两点之间的距离，二维的公式是 $d=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$；三维的公式是 $d=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2}$。

如图 3-10 所示，在二维空间中，给定非格上的向量，很容易找到格上的向量与其距离最近。但是，当基向量和维度增加时，寻找最近向量将变得很困难。

定义 3-2(最短向量问题)　对于给定的格 L，找到一个非零的格向量 $\boldsymbol{v}$，使得对于任意

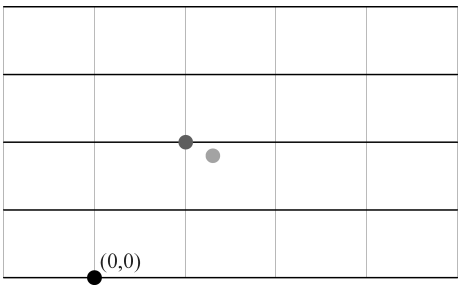

图 3-10　二维空间的格中的最近向量示例

的非零向量 $\boldsymbol{u}\in L$，有 $\|\boldsymbol{v}\|\leqslant\|\boldsymbol{u}\|$。

换种说法，该问题试图找到格 L 上一个最短的向量 $\boldsymbol{v}\in L$，其与零点的欧氏距离最小。通常使用符号 $\|\boldsymbol{v}\|$ 来表示向量 $\boldsymbol{v}$ 与零点的欧氏距离。

同最近向量问题一样，在基向量和维度够大的情况下，寻找最短向量是一个难题。解决这些问题的难度在很大程度上取决于其对应的格的基的性质。若格的基尽可能相互正交，则存在多项式时间内解决 SVP 和 CVP 的方法。若格的基的正交程度很差，则目前解决 SVP 和 CVP 的最快算法也需要指数级的计算时间。

因此，通过将正交程度差的基作为公钥，将正交程度好的基作为私钥，将解密系统设计为解决格上的最短向量问题或者最近向量问题，就能够提供一种基于格的加密方案。以最近向量问题为例，可以将数据编码到一个安全的格点，加密算法会生成一个离它近的密文，解密的时候拥有私钥的一方可以在多项式时间内找到密文对应的最近向量，但是拥有公钥和其他公开参数的一方在多项式时间内是无法完成求解的。

注意：在这里，也可以提前解释全同态加密的自举操作，因为加密后的密文是一个带噪声的点，这些点如果反复做乘法等运算，会让噪声越来越大，产生过大偏差后会让最近或者最短的格点无法正确求解出来，这时就需要自举运算来消除噪声。

基于格的密码系统具有以下两个优点。

(1) 使用线性代数操作实现加解密，具有易实现、高效率的特点。

(2) 安全性高。为达到 k 个位的安全等级，传统基于大数分解或离散对数问题的加密系统的加解密操作需要 $O(k^3)$ 的时间复杂度，而基于格的加密系统仅需要 $O(k^2)$ 的时间复杂度。随着量子计算机的问世，大数因式分解之类的经典难题可以在多项式时间被解决，但是量子计算机尚无法在多项式时间内解决格所对应的难题。

4. 理想格

循环格是一种特殊的格，循环格最显著的优点就是能够用一个向量来表示，可以采用相关算法来加速运算，可以进一步解决基于一般格上的密码方案中密钥量大、运行效率较低的问题。对于一系列向量 $\boldsymbol{v}_0=(v_1,v_2,v_3,\cdots,v_n)^{\mathrm{T}}$，$\boldsymbol{v}_1=(v_n,v_1,v_2,\cdots,v_{n-1})^{\mathrm{T}}$，$\boldsymbol{v}_2=(v_{n-1},v_n,v_1,\cdots,v_{n-2})^{\mathrm{T}}$，$\cdots\boldsymbol{v}_n=(v_2,v_3,v_4,\cdots,v_1)^{\mathrm{T}}$，以循环生成的 n 个向量为基生成的格被称为循环格。

理想格是对循环格概念的推广，一般格是群的子群，理想格指该格同时也是环上的理想。在多项式环 $Z[x]/(f(x))$ 上，循环格的基是通过给出一个多项式 $\boldsymbol{v}\in L$，然后对其连续模乘 x 得到 $\{\boldsymbol{v}_i=\boldsymbol{v}_0\cdot x^i(\mathrm{mod}f(x))\mid i\in[0,n-1]\}$。可以证明，在多项式环上构建的循环

格即为环的理想，也称理想格。

理想格具有以下两个优点。

(1) 可以降低格表示的空间尺寸。格的表示方式需要比较大的空间，比如，用一个 $n \times n$ 矩阵来表示一组基，则需要存储 n^2 个元素。而理想格的表示则非常简单，对于基而言，给出一个多项式即可。

(2) 理想格具有理想的特性，即对内具有乘法封闭性，对外具有乘法吸收性，这一特点使得理想格很容易构造全同态加密方案。

3.4.2 Gentry 方案

尽管有很多关于部分同态加密和类同态加密的方案，然而在同态加密概念提出后的 30 年间，并没有真正能够支持无限制的各类同态操作的全同态加密方案问世。直到 2009 年，斯坦福大学的博士生 Gentry 在他的论文中提出了第一个切实可行的全同态加密方案[①]。

1. 自举操作

类同态加密方案可以支持有限次数的各类同态操作，不满足强同态。如果想要不断地进行同态操作，一种简单直接的方法是将该密文解密并且再次加密，从而能够获得一个全新的密文，这个过程简称为刷新。刷新之后的密文相当于被重置回刚刚加密的状态，从而继续支持更多的同态操作。但是这样的话，需要使用密钥对密文进行解密，这违背了同态加密在加密状态下进行持续运算的原则。

Gentry 敏锐地察觉到，如果能够设计一种加密方案，它的解密操作本身能够做成同态操作，就能够在全程不解密的情况下完成刷新操作，这个操作称为自举。

自举过程可简述如下。

(1) 对于给定的同态加密方案 ε。在生成公私钥对(sk，pk)之后，用公钥 pk 加密私钥 sk 得到$\overline{\text{sk}}$。

(2) 在对密文 ct 进行同态加密之前，应用公钥 pk 再次加密得到两次加密的密文$\overline{\text{ct}}$。设解密操作为 D_ε，其中：ε 支持同态解密，使用密文$\overline{\text{ct}}$和密钥$\overline{\text{sk}}$进行同态解密 Evaluate(pk，D_ε，$\overline{\text{ct}}$，$\overline{\text{sk}}$)。此时，更早被加密的密文已经被解密，生成密文为此轮刚加密的全新密文。

(3) 在新密文上执行一系列同态操作。

通过自举技术可以将原来仅支持有限次同态操作的近似同态加密方案改造成支持无限次同态操作的全同态方案。基于这一思想，Gentry 提出了基于理想格的全同态加密方案。

2. 近似同态加密方案

1) 具体实现

Gentry 的基于理想格的近似同态加密方案的具体实现如下。

(1) 密钥生成。

给定一个多项式环 $R=Z[x]/(f(x))$，R 上的一个理想 I 及其固定基 B_I，通过循环格生成理想格 J，满足 $I+J=R$，生成两组 J 的基(B_J^{sk}，B_J^{pk})作为公私钥对。其中：私钥 B_J^{sk}

① GENTRY C. Fully homomorphic encryption using ideal lattices[C]//Proceedings of the forty-first annual ACM symposium on Theory of computing, 2009: 169-178.

正交化程度较高；公钥 B_J^{pk} 正交化程度较低。另外，提供一个随机函数 $\mathrm{Samp}(B_I,x)$用于从 $x+B_I$ 的陪集中抽样。最终，$(R,B_I,B_J^{pk},\mathrm{Samp}())$为公钥，$B_J^{sk}$ 为私钥。

注意：$I+J=R$ 表示 I 和 J 上的元素，运算＋的结果在 R 上。

(2) 加密。

通过函数 $\mathrm{Samp}(B_I,x)$随机选择向量 $\boldsymbol{r},\boldsymbol{g}$，使用 B_J^{pk} 对明文 $\boldsymbol{m}\in\{0,1\}^n$ 进行加密，有

$$\boldsymbol{c}=\mathrm{Enc}(\boldsymbol{m})=\boldsymbol{m}+\boldsymbol{r}\cdot B_I+\boldsymbol{g}\cdot B_J^{pk}$$

其中，$\boldsymbol{g}\cdot B_J^{pk}$ 是理想格 J 上的一个元素。

(3) 解密。

通过私钥 B_J^{sk} 解密密文 $\boldsymbol{c}$，得到

$$\boldsymbol{m}=\boldsymbol{c}-B_J^{sk}\cdot\lfloor(B_J^{sk})^{-1}\cdot\boldsymbol{c}\rceil(\mathrm{mod}B_I)$$

其中，$\lfloor\ \rceil$表示对向量各维度坐标进行四舍五入取整。

2) 正确性

在加密阶段，密文 $\boldsymbol{c}$ 可以看作一个格 J 中的元素 $\boldsymbol{g}\cdot B_J^{pk}$ 加上噪声 $\boldsymbol{m}+\boldsymbol{r}\cdot B_I$。

在解密阶段，$B_J^{sk}\cdot\lfloor(B_J^{sk})^{-1}\cdot\boldsymbol{c}\rceil$为应用取整估计法求解最近向量问题，即找到密文向量在格 J 中最近的向量，即 $B_J^{sk}\cdot\lfloor(B_J^{sk})^{-1}\cdot\boldsymbol{c}\rceil=\boldsymbol{g}\cdot B_J^{pk}$。因此，有

$$\boldsymbol{m}=\boldsymbol{c}-B_J^{sk}\cdot\lfloor(B_J^{sk})^{-1}\cdot\boldsymbol{c}\rceil(\mathrm{mod}B_I)=\boldsymbol{c}-\boldsymbol{g}\cdot B_J^{pk}(\mathrm{mod}B_I)=\boldsymbol{m}+\boldsymbol{r}\cdot B_I(\mathrm{mod}B_I)$$

注意：应用取整估计法解最近向量问题要求 $\boldsymbol{m}+\boldsymbol{r}\cdot B_I$ 足够小，才能保证其加密的格元素 $\boldsymbol{g}\cdot B_J^{pk}$ 和解密时找到的最近的格元素是相同元素，即 $B_J^{sk}\cdot\lfloor(B_J^{sk})^{-1}\cdot\boldsymbol{c}\rceil=\boldsymbol{g}\cdot B_J^{pk}$。

3) 安全性

上述加解密过程，安全性规约到最近向量问题的求解上，只有在格的基正交程度较高的私钥上可以获得尽可能接近的格向量，正交程度较低的基 B_J^{pk} 无法解密明文。

4) 同态性

在该方案中，明文和密文之间的线性关系使得同态操作易于实现，直接密文相加即可实现同态加法，即

$$\boldsymbol{c}_1+\boldsymbol{c}_2=\boldsymbol{m}_1+\boldsymbol{m}_2+(\boldsymbol{r}_1+\boldsymbol{r}_2)\cdot B_I+(\boldsymbol{g}_1+\boldsymbol{g}_2)\cdot B_J^{pk}$$

结果仍在密文空间中，并且只要 $\boldsymbol{m}_1+\boldsymbol{m}_2+(\boldsymbol{r}_1+\boldsymbol{r}_2)\cdot B_I$ 相对较小，即可运用上述解密方法得到明文 $\boldsymbol{m}_1+\boldsymbol{m}_2$。其同态乘法也可以直接使用密文相乘，即

$$\boldsymbol{c}_1\cdot\boldsymbol{c}_2=\boldsymbol{e}_1\boldsymbol{e}_2+(\boldsymbol{e}_1\boldsymbol{g}_2+\boldsymbol{e}_2\boldsymbol{g}_1+\boldsymbol{g}_1\boldsymbol{g}_2)\cdot B_J^{pk}$$

其中：$\boldsymbol{e}_1=\boldsymbol{m}_1+\boldsymbol{r}_1\cdot B_I$；$\boldsymbol{e}_2=\boldsymbol{m}_2+\boldsymbol{r}_2\cdot B_I$。该结果仍然在密文空间中，并且当$|\boldsymbol{e}_1\cdot\boldsymbol{e}_2|$足够小时，可以通过上述解密方法获得 $\boldsymbol{m}_1\cdot\boldsymbol{m}_2$。

5) 自举

随着加法同态和乘法同态的积累，密文中的噪声项逐渐积累增大，直至无法从密文中解密明文，这时就需要借助自举技术消除噪声，使其支持无限次数的加法和同态乘法。因为 Gentry 自举技术较为复杂，这里不详细介绍。

3.4.3　CKKS 算法[①]

1. 容错学习

容错学习是在格的难题上构建出来的问题，可以看作解一个带噪声的线性方程组：给

① BRAKERSKI Z, VAIKUNTANATHAN V. Efficient fully homomorphic encryption from (Standard) LWE [C]//Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science, 2011: 97-106.