文章编号:1671-4229(2021)04-0029-17

LWE 问题的分析策略及格基约减算法综述

王镭璋[1]，张帅领[2]，王保仓[1]
(1. 西安电子科技大学 综合业务网理论及关键技术国家重点实验室，陕西 西安　710071；
2. 中国电子科技集团公司 第三十研究所，四川 成都　610041)

摘　要：格公钥密码因其数学困难问题的平均/最坏复杂度等价性、代数结构的线性性和丰富的密码等功能而被普遍认为是最有发展前途的后量子密码。目前有大量格密码算法的安全性基于格上的错误学习(LWE)问题而设计。因此,研究 LWE 的求解算法对深刻理解格公钥密码算法的安全性至关重要。文章首先针对不同类型 LWE 问题的不同求解策略进行梳理和比较;其次,归纳整理了目前已有的格基约减算法的基本思想和技术原理;最后,对基于格归约的方法求解 LWE 问题的研究重点和趋势作了展望。

关键词：格密码；带错误的学习问题；格基归约算法
中图分类号：TP 309.7　　**文献标志码**：A

Analysis strategies and solving algorithms of the LWE problem: A survey

WANG Lei-zhang[1], *ZHANG Shuai-ling*[2], *WANG Bao-cang*[1]
(1. State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China;
2. The 30th Research Institute of China Electronic Science & Technology Group Inc., Chengdu 610041, China)

Abstract: Lattice public key cryptography is widely conceived as the most promising candidate in the quantum computing era due to the average-case/hard-case equivalence of the underlying mathematically intractable problems, the linearity of the involved algebraic structures, and the tremendous cryptographic functionalities. At present, numerous post-quantum lattice cryptographic algorithms were developed to rely their securities on a lattice problem: the learning with errors (LWE) problem. Therefore, it is of vital significance to study the algorithms for solving the LWE problems to give insights into the security of lattice public key cryptography. This paper firstly reviews and compares different solution strategies for distinct types of LWE problems. Secondly, we outline the basic ideas and technical principles of the existing lattice reduction algorithms. Finally, some prospections are made on the research focuses and trends in solving the LWE problems with lattice based reduction algorithms.

Key words: lattice cryptography; learning with error; lattice basis reduce algorithm

由于格密码具有抗量子、高效率、最坏情况/平均情况等价性和丰富的密码功能等特点引起了学术界的广泛关注。第一个基于格的加密方案是由 Ajtai 等[1]提出的。后来 Regev[2]对该方案进行了简化和改进。Regev 工作的主要成果之一就是引入了带错误学习问题(Learning With Errors, LWE),该问题在密码构造中使用相对简单,并且至少同最坏情况下 SVP 的变体问题一样难[2]。目前,LWE 问题已被证明是加密应用程序的通用构建模块([Regev05][2]、[GPV08][3]、[BV11][4]和[ABB10][5])。因此,对 LWE 问题求解算法的深入分析对深刻理解格密码安全性非常重要[6]。

作者简介：王镭璋(1996—),男,博士研究生. E-mail:lzwang_2@stu.xidian.edu.cn

引文格式：王镭璋，张帅领，王保仓. LWE 问题的分析策略及格基约减算法综述[J]. 广州大学学报(自然科学版),2021,20(4):29-45.

对LWE问题的一个标准求解做法就是用Kannan嵌入[7]将其归约到某个格上的唯一最短向量问题(unique Shortest Vector Problem, u-SVP)。而求解格上最短向量问题的主流做法是采用格基约减算法进行求解。格基约减算法作为格密码分析研究的核心内容,其中对格基约减算法的预测和优化仍是目前一个活跃的研究领域[8]。虽然已有的研究在理论上确实给出了格基约减算法在最坏情况下性能的边界定理[9-18],但这些边界是渐近的,在应用于实际甚至密码实例时不一定严格[11,14,19]。对这些算法行为的预测依赖于启发式和近似,其中一些预测已知在某些特定的范围内会失效[11,20]。随着格基约减算法的标准化,对其性能的精确估计也就成为一个关键的问题。

1 预备知识

定义1 格:已知$\mathbb{R}^m$中的n个线性无关的列向量组$\boldsymbol{B}=\{\boldsymbol{b}_1,\cdots,\boldsymbol{b}_n\}$($m\geqslant n$),这组向量的所有整系数线性组合的集合称为格$L$,记为

$$L(\boldsymbol{B})=\{\sum_{i=1}^{n}x_i\mathbf{b}_i,x_i\in\mathbb{Z}\},$$

其中,线性无关的向量组$\boldsymbol{B}$称为格L的一组基。n为格的秩,m为格的维数。

定义2 最短向量问题(SVP):给定$\mathbb{R}^m$中秩为n的格L的一组基$\boldsymbol{B}$,寻找一个非零向量$u\in L(\boldsymbol{B})$,使得u是$L(\boldsymbol{B})$是非零最短向量,即u满足$\|u\|=\lambda_1(L)$。

定义3 LWE分布[2]:给定正整数n和q,s是$\mathbb{Z}_q^n$中的向量。从$\mathbb{Z}_q^n$中均匀随机地抽取向量a,从标准差为$\sigma=q\alpha$(其中,$0<\alpha<1$表示错误率)的$\mathbb{Z}$上离散高斯分布中随机抽取整数e,计算$(a,b)=(a,\langle a,s\rangle+e)\in\mathbb{Z}_q^n\times\mathbb{Z}_q$,并用$L_{s,\chi}$表示$(\boldsymbol{a},b)$在$\mathbb{Z}_q^n\times\mathbb{Z}_q$上的概率分布。

定义4 LWE问题[8]:判定版本的LWE问题(Decision-LWE)可描述为判断一组LWE随机实例中的样本$(\boldsymbol{a},b)\in\mathbb{Z}_q^n\times\mathbb{Z}_q$是来自分布$L_{s,\chi}$还是来自$\mathbb{Z}_q^n\times\mathbb{Z}_q$上的均匀分布。计算版本的LWE问题(Search-LWE)可描述为根据来自分布$L_{s,\chi}$的一组实例中的样本$(\boldsymbol{a},b)\in\mathbb{Z}_q^n\times\mathbb{Z}_q$,计算$s$。

一般将一组LWE的实例写作矩阵的形式$(\boldsymbol{A},\boldsymbol{b})=(\boldsymbol{A},\boldsymbol{As}+\boldsymbol{e})\in\mathbb{Z}_q^{m\times n}\times\mathbb{Z}_q^n$,其中,矩阵$\boldsymbol{A}$的每一行是一个服从分布$L_{s,\chi}$选取的LWE实例$(\boldsymbol{a},b)=(\boldsymbol{a},\langle\boldsymbol{a},\boldsymbol{s}\rangle+e)\in\mathbb{Z}_q^n\times\mathbb{Z}_q$,$m$是实例的个数。此外,根据LWE实例中秘密向量$\boldsymbol{s}$分布的不同可将LWE问题的类型进行划分。若$\boldsymbol{s}\xleftarrow{\$}\mathbb{Z}_q^n$,就是标准型LWE;若$\boldsymbol{s}\xleftarrow{\$}\chi_\sigma^n$,就是正规型LWE(其中$\chi_\sigma$是$\mathbb{Z}$上标准差为$\sigma$的离散高斯分布);若$\boldsymbol{s}\xleftarrow{\$}\{0,1\}^n$或$\{-1,0,1\}^n$,就是二元或三元(binary/ternary)LWE。

2 LWE问题的分析策略以及非基于格的求解算法

2.1 LWE问题的分析策略

目前,对于不同类型LWE问题的分析方法主要分为4种策略:SIS策略、BDD策略、ISIS策略和直接求解(Direct)策略,见图1。

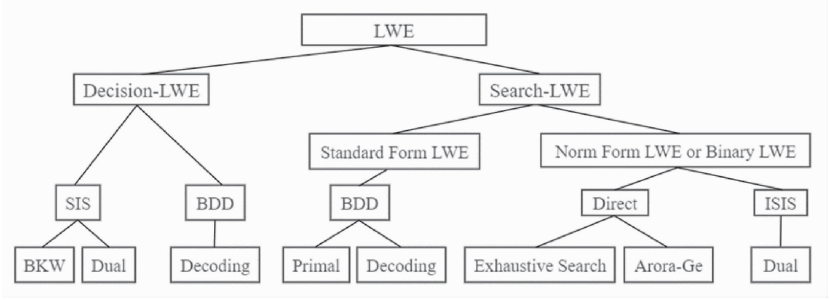

图1 LWE分析策略及方法

Fig. 1 LWE analysis strategies and methods

对于不同策略下的求解算法主要有4种:组合算法、代数算法、基于格的原始攻击和基于格的对偶攻击。非基于格的求解算法就是组合算法和代数算法2种。组合方法一般指BKW算法[21],

该算法可看作一种扩展的高斯消元法,一次处理多个元素,这种算法最大的问题是需要指数多的样本。代数方法是 Arora-Ge 算法[22],该方法是一种非线性化方法,同样需要指数多的样本。思想就是将 LWE 实例分成单项式相乘的形式,从而通过求解单项式的根将秘密向量 $\boldsymbol{s}$ 各个分量恢复出来。算法关于维数是指数的,对于错误向量较小的情况可以降到亚指数级别。下面对组合方法和代数方法分别进行详细描述。

2.2　非基于格的求解算法

2.2.1　组合算法

组合算法的代表算法是 BKW 算法,这是一种特殊的对偶攻击方法,最早由 Avrim Blum, Adam Kalai 和 Hal Wasserman 提出用来求解 LPN 问题,该算法能够以亚指数的时间复杂度 $2^{O(n/\log n)}$ 求解 LPN 问题。由于 LWE 问题可看做 LPN 问题从 $\mathbb{Z}_2$ 到 $\mathbb{Z}_q$ 上的扩展,因此,扩展版本的 BKW 算法适用于分析不限制样本量的 LWE 类问题,基本思路是通过分块碰撞的思想,借助广义生日攻击的方法在指数多的样本中挑选出部分样本使其加和为 0,此时零向量关于样本的系数向量即为对偶格中的短向量,由本节中介绍的对偶攻击可知,这些对偶格中的短向量可以用作 LWE 实例的区分器。

具体来说, BKW 构造短向量 v_i 的方法如下:给定一组样本 $\mathbb{Z}_2$,BKW 分解行向量的 n 个分量为 a 个块,每个块的宽度为 $b = n/a$。算法共分成 a 个阶段,第 i 个阶段通过充分多的上一级样本碰撞(加法运算)获得第 i 块元素全零的样本。假设初始样本数量为 $2^a q^b$,在最优的情况下,算法结束时可获得 q^b 个长度为 $\sqrt{2^a}$ 的短向量,使用这些短向量做区分攻击的方法与对偶攻击中的区分方法类似,即每个短向量的区分优势为 $\varepsilon = \exp[-2\pi^2 2^a (\sigma/q)^2]$,大约需要 $1/\varepsilon^2$ 个短向量来获得不可忽略的区分成功率,因此,BKW 需要通过调整分块方式来平衡区分优势与样本数量以获得最优的算法复杂度。

BKW 算法的主要改进技术包括: Albrecht 等[23]在 PKC 2014 上提出的 Coded-BKW 算法中求解小密钥版本的 LWE 问题所采用的 Lazy Modulus Switching 技术;2015 年,文献[24 - 25]分别独立提出的通过使用不同大小的分块平衡 BKW 算法复杂度的技术;2017 年,Guo 等[26]提出的在 BKW 算法中结合筛法的改进技术,该算法是当前具有最优时间复杂度的 BKW 类算法。

2.2.2　代数攻击——Arora-Ge 算法

代数攻击最早由 Arora 等[22]提出,适用于不限制样本量的 LWE 类问题,它的基本思路是:构建以错误 e 为根的无噪音非线性多项式方程组,通过代换所有单项式构成线性方程组并求解。具体来说,给定 LWE 实例矩阵 $(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{As} + \boldsymbol{e})$,根据错误分布(高斯分布和小均匀分布等)的性质可知错误分布 $\boldsymbol{e}$ 在有界区间 $[-t, t]$ 中 $(t \in \mathbb{Z}, d = 2t + 1 < q)$。则可构造多项式 $p(x) = x\prod_{i=1}^{t}(x + i)(x - i)$,显然 e 是多项式 $p(x)$ 的一个根,因此,s 是 $p(ax—b)$ 的一个根,可通过求解上述多项式的方式来解决对应的 LWE 问题。在文献[22]中,高次多项式方程组是通过替换单项式获得线性方程组求解。在改进算法中[27],则用 Grobner 基来求解方程组,这一改进算法所需要的方程数量更少,但求解方程的时间复杂度更高。

由于 BKW 算法和 Arora-Ge 算法需要指数级别的实例数,而现实中的格基加密方案至多泄露关于 LWE 实例维数 n 的多项式多个实例([Regev05]型加密方案至多泄露 $n\log q$ 个实例,[LP11]型加密方案[28]至多泄露 $2n$ 个实例)。因此,针对实际方案的攻击,以上 2 种求解算法均无法实施。针对实际方案的攻击一般使用的是基于格的攻击方法。

3　基于格的求解方法

针对判定性 LWE 问题采用 SIS 策略,归约到求正交格上短向量。针对搜索型 LWE 问题,根据其秘密向量 $\boldsymbol{s}$ 的模长大小采用不同的策略。如果是标准型(Standard form) LWE 问题则采用使用 BDD 策略的原始攻击或是 Decoding 方法;如果是 Normal Form LWE 或是 Binary LWE 应当充分利用秘密向量 $\boldsymbol{s}$ 的模长很短这一信息,从而应该使用 I-SIS 策略的对偶攻击。

3.1　判定性 LWE 问题

3.1.1　SIS 策略

针对判定性 LWE 问题,区分攻击采用 SIS 策略。SIS 策略是将判定性 LWE 问题归约到由矩阵 $\boldsymbol{A}$ 定义的 q-ary 正交格 $\Lambda_q^{\perp}(\boldsymbol{A})$ 上的 SIS 问题。具

体来说就是考虑到如果判别器拿到的是 LWE 实例$(\boldsymbol{A},\boldsymbol{b})\in\mathbb{Z}_q^{m\times n}\times\mathbb{Z}_q^n$有:$\boldsymbol{b}=\boldsymbol{As}+\boldsymbol{e}$,且噪声向量$\boldsymbol{e}$是一个短向量。在由$\boldsymbol{A}$生成的正交格$\Lambda_q^{\perp}(\boldsymbol{A})=\{\boldsymbol{x}\in\mathbb{Z}_q^m\mid\boldsymbol{xA}=0(\mathrm{mod}q)\}$上找到向量$\boldsymbol{v}$,由于$\boldsymbol{v}\in\Lambda_q^{\perp}(\boldsymbol{A})$有:$\langle\boldsymbol{v},\boldsymbol{b}\rangle=\boldsymbol{vAs}+\langle\boldsymbol{v},\boldsymbol{e}\rangle=\langle\boldsymbol{v},\boldsymbol{e}\rangle$,那么只要找到的向量$\boldsymbol{v}$足够的短,$\langle\boldsymbol{v},\boldsymbol{b}\rangle$会是一个很小的数。如果判别器拿到的实例$\boldsymbol{b}$来自$\mathbb{Z}_q^n$上均匀分布,那么$\langle\boldsymbol{v},\boldsymbol{b}\rangle$在$\mathbb{Z}_q^n$上也是均匀随机的。如此就将判定性 LWE 问题转化为求$\Lambda_q^{\perp}(\boldsymbol{A})$格上短向量问题。[LP11][28]给出了该策略区分优势的表达式为$\varepsilon=\exp(-\pi\cdot(\|\boldsymbol{v}\|\cdot s/q)^2)$,当$\|\boldsymbol{v}\|=4.79q/s$时,区分优势约为$2^{-72}$。然而要以高概率成功判别 LWE 实例,需要$\|\boldsymbol{v}\|\leqslant q/(2s)$。这就需要格基质量足够地好。由于对偶格与$\Lambda_q^{\perp}(\boldsymbol{A})$只差一个模数$q$,所以区分攻击也称为 Dual 攻击。[LP11][28]指出直接将判定性 LWE 问题视为 BDD 问题并用 Decoding 方法求解可以在给定相同质量格基条件下以更高的概率求解判定性 LWE 问题。下面介绍使用 Decoding 方法求解判定性 LWE 问题。

3.1.2 BDD 策略

[LP11][28]中的 Decoding 策略强于区分攻击,因为 Decoding 策略直接将噪声向量$\boldsymbol{e}$恢复出来,实际也可以视为是求解 Search-LWE 问题的一种方法。但该策略在使用比区分攻击质量更差的格基时仍有着一样甚至更好的优势。其基本思想是将 LWE 实例中将$\boldsymbol{b}$视作$\Lambda_q(\boldsymbol{A})$格上 BDD 问题的目标向量,然后直接利用最近平面算法找到距离$\boldsymbol{b}$最近的格点$\boldsymbol{As}$。该策略成功当且仅当错误向量$\boldsymbol{e}$落到$\boldsymbol{P}_{1/2}(\boldsymbol{B}^*)$内下面给出直接使用 Babai 算法[29]找到格点 As 的概率估计:

$$\Pr[\boldsymbol{e}\in\boldsymbol{P}(\boldsymbol{B}^*)]=\prod_{i=1}^{m}\Pr[|\langle\boldsymbol{e},\boldsymbol{b}_i^*\rangle|<\|\boldsymbol{b}_i^*\|^2/2]=\prod_{i=1}^{m}erf\left(\frac{\|\boldsymbol{b}_i^*\|\cdot\sqrt{\pi}}{2\alpha q}\right),$$

其中,$erf(x)=(2/\sqrt{\pi})\int_0^x e^{-t^2}\mathrm{d}t$。可以看到如果使用区分攻击,求解短向量$\|\boldsymbol{v}\|\approx 4q/s$,区分攻击才有$2^{-72}$的区分优势。

而 Decoding 策略使用最简单的 Babai 算法,只要当$\|\boldsymbol{b}_m^*\|=q/\|\boldsymbol{v}\|=s/4$,就有$erf(\sqrt{\pi}/8)\approx 0.25$概率求解出距离$\boldsymbol{b}$最近的格点。可以看到 Decoding 策略的优势优于区分攻击。

由上述条件可知该策略最大的问题同样需要一个"好"格基。其实可以直观理解这一点:错误向量$\boldsymbol{e}$服从高斯分布的,而由几何级数假设我们知道$\|\boldsymbol{b}_i^*\|$是递减的,$P_{1/2}(B^*)$的形状远不如错误向量$\boldsymbol{e}$的分布对称。当格基较差时,错误向量$\boldsymbol{e}$不太可能落到$P_{1/2}(B^*)$中。为解决这个问题,文献[28]提出了一种多次递归调用 Babai 算法的改进最近平面算法。

LP 最近平面算法,[LP11][28]中的优化最近平面算法在每一次执行寻找当前距离目标向量最近的超平面操作时,从原先的只选距离目标向量最近的超平面改成:选择d_i个超平面$(i\in\mathbb{Z}^+)$,如此将$P_{1/2}(B^*)$往每个$\boldsymbol{b}_i^*$方向上扩大为原来的d_i倍,使得$\boldsymbol{e}$落入$\mathrm{P}_{1/2}(\mathrm{B}^*)$的概率增大。同时,新算法要记录下全部$\prod_{i=1}^{m}d_i$个候选值,并分别计算它们到目标向量$\boldsymbol{b}$的距离,取其中距离最小的作为其优化最近平面算法输出结果。如此在付出$\prod_{i=1}^{m}d_i$次对$(\boldsymbol{B},\boldsymbol{b})$的 Babai 算法的计算开销下,算法的成功率增大为

$$\Pr[\boldsymbol{e}\in\boldsymbol{P}(\boldsymbol{B}^*)]=\prod_{i=1}^{m}\Pr[|\langle\boldsymbol{e},\boldsymbol{b}_i^*\rangle|<\|\boldsymbol{b}_i^*\|^2/2]=\prod_{i=1}^{m}erf(d_i\cdot\|\boldsymbol{b}_i^*\|\cdot\sqrt{\pi}/(2\alpha q))。$$

3.2 搜索型 LWE 问题

3.2.1 BDD 策略 - 原始攻击

针对搜索型 LWE 问题,BDD 策略先是将搜索型 LWE 问题归约到某个q-ary 格上的 BDD 问题,再利用 Kannan 嵌入[7]将q-ary 格上的 BDD 问题归约到最终构造格上的 uSVP 进行求解。根据 LWE 问题中s的分布不同,最终构造的格基的形式有所不同。

最早提出该策略的是文献[28],特别地,给定分布$L_{s,\chi}$通过采样获得实例$(\boldsymbol{A},\boldsymbol{b})\in\mathbb{Z}_q^{m\times n}\times\mathbb{Z}_q^n$满足:

$$\boldsymbol{A}=\begin{pmatrix}\boldsymbol{A}_1\\ \boldsymbol{A}_2\end{pmatrix},\boldsymbol{As}+\boldsymbol{e}=\boldsymbol{b}(\mathrm{mod}q),$$

其中,$\boldsymbol{A}_2$是由均匀采样获得,$\boldsymbol{e}$来源于错误分布χ,另外$\boldsymbol{A}_1\in\mathbb{Z}_q^{n\times n}$,$n$阶模$q$下可逆方阵。在文献[28]中通过不断地从分布$L_{s,\chi}$中抽取生成新的实例直到这些实例形成可逆方阵$\boldsymbol{A}_1$,并直接丢掉其余实例。易知:

$$\begin{pmatrix}\boldsymbol{b}_1\\ \boldsymbol{b}_2\end{pmatrix}=\begin{pmatrix}\boldsymbol{A}_1\\ \boldsymbol{A}_2\end{pmatrix}\boldsymbol{s}+\begin{pmatrix}\boldsymbol{e}_1\\ \boldsymbol{e}_2\end{pmatrix},$$

由于 $\boldsymbol{A}_1\in\mathbb{Z}_q^{n\times n}$ 模 q 可逆,可以由初始的 LWE 问题实例$(\boldsymbol{A},\boldsymbol{s})$构造一个新的 LWE 实例$(\overline{\boldsymbol{A}},\overline{\boldsymbol{t}})$:$\overline{\boldsymbol{t}}=\overline{\boldsymbol{A}}\boldsymbol{b}_1+\boldsymbol{b}_2$,其中,$\overline{\boldsymbol{A}}=-\boldsymbol{A}_2\boldsymbol{A}_1^{-1}$。容易看出:$\overline{\boldsymbol{t}}=\overline{\boldsymbol{A}}\boldsymbol{e}_1+\boldsymbol{e}_2$。如此就将 LWE 问题由求解 $(\boldsymbol{s},\boldsymbol{e})$ 变成了求解 $(\boldsymbol{e}_1,\boldsymbol{e}_2)$,所以将求解任意分布秘密向量 $\boldsymbol{s}$ 的 LWE 问题转变为离散高斯分布上秘密向量的 LWE 问题。然后可以将这个新的 LWE 问题归约到 BDD 问题,再利用 Kannan 嵌入归约到构造格上的 u-SVP。最后用格基约化算法去求解 uSVP。特别地,其构造格基 $\boldsymbol{M}$ 为

$$\boldsymbol{M}=\begin{pmatrix}\boldsymbol{I}_n & \overline{\boldsymbol{A}} & 0\\ 0 & q\boldsymbol{I}_{m-n} & 0\\ 0 & \overline{\boldsymbol{t}} & 1\end{pmatrix},$$

可见,存在向量 $\boldsymbol{u}^{\mathrm{T}}=(\boldsymbol{e}_1^{\mathrm{T}}\quad\boldsymbol{e}_2^{\mathrm{T}})$ 使得:

$$\boldsymbol{v}=\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}=(\boldsymbol{e}_1^{\mathrm{T}},\boldsymbol{e}_2^{\mathrm{T}},1)\in\Lambda(\boldsymbol{M})$$

该策略的另外一个格基构造由 Albrecht 等[30]提出。具体来说,该策略考虑到 $\boldsymbol{As}\approx\boldsymbol{b}$,且 $\Lambda_q(\boldsymbol{A})$ 格上格点 $\boldsymbol{As}$ 到目标向量 $\boldsymbol{b}$ 的距离不超过 $\|\boldsymbol{e}\|<\sigma\sqrt{m}$,所以自然将 LWE 问题归约到 $\Lambda_q(\boldsymbol{A})$ 格上 BDD 问题。最后在利用 Kannan 嵌入将目标向量 $(\boldsymbol{b},1)$ 嵌入到 $\Lambda_q(\boldsymbol{A})$ 格上,转化为最终构造格基 $\boldsymbol{M}$ 上 u-SVP 的求解。这个归约过程可以总结为如下形式,Search-LWE≤BDD≤u-SVP,其中,第二个不等号处使用了 Kannan 嵌入,此外实验结果表明[31],选取 Kannan 嵌入因子 $t=1$ 可使最终求解 u-SVP 时更快,故 t 一般取 1。

其格基矩阵 $\boldsymbol{M}$ 为

$$\boldsymbol{M}=\begin{pmatrix}\boldsymbol{I}_n & \boldsymbol{A}' & 0\\ 0 & q\boldsymbol{I}_{m-n} & 0\\ \boldsymbol{b}^{\mathrm{T}} & \cdots & t\end{pmatrix}。$$

如此格 $\boldsymbol{M}$ 上存在一个不同寻常的短向量 $(\boldsymbol{e},1)$,$\|\boldsymbol{e}\|<\sigma\sqrt{m}$,如此完成 LWE 问题到构造的格 $\boldsymbol{M}$ 上的 uSVP 的归约。

以上提到 2 种格攻击一般称为原始攻击,其成功条件(08 估计[10]见 3.2.3 节) 是相同的,即 2 种格基在求解标准型 LWE 问题本质上没有区别。

3.2.2　ISIS 策略 - 对偶攻击

针对秘密向量 $\boldsymbol{s}$ 很短的正规型 LWE 问题,以及二进制型 LWE 问题,除了使用 BDD 策略的原始攻击,[BG14][31]提出了一种更高效的求解策略:将 Normal Form(Binary)型 LWE 问题视为 ISIS 问题进行求解,充分利用了 s 也很短这一信息。一般将使用 ISIS 策略的[BG14]型攻击称为对偶攻击[31]。归约的思路可以概况如下:

Normal Form(Binary) LWE≤ISIS≤BDD≤u-SVP,

其中,第三个不等号处使用了 Kannan 嵌入。更详细的过程如下:

给定服从分布 $L_{s,\chi}$ 选取的 m 个 LWE 实例$(\boldsymbol{A},\boldsymbol{b})\in\mathbb{Z}_q^{m\times n}\times\mathbb{Z}_q^n$,考虑 ISIS 实例:

$$(\boldsymbol{A}\,|\,\boldsymbol{I}_m)\begin{pmatrix}\boldsymbol{s}\\ \boldsymbol{e}\end{pmatrix}=\boldsymbol{b}(\mathrm{mod}q)。$$

下一步就是将这个 ISIS 实例归约到 $\Lambda_q^{\perp}(\boldsymbol{A}')$ 上 BDD。显然上述 ISIS 问题有特解 $\boldsymbol{w}^{\mathrm{T}}=(0,\boldsymbol{b}^{\mathrm{T}})$,那么只需求出 $\Lambda_q^{\perp}(\boldsymbol{A}')$ 上距离特解 $\boldsymbol{w}$ 最近的格向量 $\boldsymbol{v}$,$\boldsymbol{w}-\boldsymbol{v}$ 就是 ISIS 问题的解。所以问题转化为 $\Lambda_q^{\perp}(\boldsymbol{A}')$ 上 BDD 问题。最后用 Kannan 嵌入 $\Lambda_q^{\perp}(\boldsymbol{A}')$ 上 BDD 问题转化为下面构造的格上 u-SVP 问题:

$$\boldsymbol{M}=\begin{pmatrix}v\boldsymbol{I}_n & -\boldsymbol{A}^{\mathrm{T}} & 0\\ 0 & q\boldsymbol{I}_m & 0\\ 0 & \boldsymbol{b}^{\mathrm{T}} & 1\end{pmatrix},$$

存在整系数线性组合 $\boldsymbol{u}=(\boldsymbol{s}\,|\,*\,|\,\boldsymbol{1})$,使得 $\boldsymbol{uM}=(\boldsymbol{s}|\boldsymbol{e}|1)$ 是 $\Lambda(\boldsymbol{M})$ 上短向量。

3.2.3　2 种攻击方式的比较

上述 2 种不同攻击策略的区别在于:Bai 等[31]的格基更有效地利用了 $\boldsymbol{s}$ 与 $\boldsymbol{e}$ 同分布时,$\boldsymbol{s}$ 也是很短的这一信息,相较于原始攻击中的格基[30],文献[31]有更弱的攻击成功条件。下面先给出求解 u-SVP 的理论条件再对 2 种攻击进行更详细的比较。

08 估计[10]根据其实验结果提出:设 gap 等于格上 λ_2 与 λ_1 的比值,对 u-SVP 可以由近似因子为 gap 大小的近似最短向量求解器求解。因而 u-SVP 的 gap 越大,意味着其求解难度越小。并给出用格基约减算法求解 u-SVP 的成功条件:$(\lambda_2/\lambda_1)\geqslant\tau\delta_0^m$,其中,$\tau\in(0,1)$ 是所使用的格基约化算法相关的常数(对 BKZ 类算法一般在 0.3 ~ 0.4 之间),采用 08 估计[10]进行评估:

原始攻击的格基的 gap 为

$$\frac{\lambda_2}{\lambda_1}\approx\frac{q^{(m-n)/m}\sqrt{m/(2\pi e)}}{\sigma\sqrt{m}}\approx q^{(m-n)/m}/\sigma。$$

而对偶攻击的格基的 gap 为

$$\frac{\lambda_2}{\lambda_1}\approx\frac{q^{m/(m+n)}\sqrt{(m+n)/2\pi e}}{\sigma\sqrt{m+n}}\approx q^{m/(m+n)}/\sigma。$$

显然对偶攻击的格基 gap 更大,$\frac{m}{m+n}>\frac{m-n}{m}$,因而对于 Normal Form LWE 和 Binary LWE,对偶攻击相较于原始攻击更高效。

此外,Bai 等[31]的格基无需实例组成的 n 阶方阵要在模 q 下求逆,因此对实例的数量没有严格的要求。但原始攻击中的格基因为需要实例组成的 n 阶方阵要在模 q 下可逆,因而只能用于求解实例个数 m 大于 n 的情况。

值得注意的是 Bai 等的格基也只能用于求解 Normal Form LWE($\boldsymbol{s}$ 与 $\boldsymbol{e}$ 同分布)问题,对 $\boldsymbol{s}$ 取自均匀分布的 Standard LWE 问题无法直接使用。而原始攻击的格基就不存在这个问题,它们的格基对 $\boldsymbol{s}$ 取任意分布的 LWE 问题都是适用的。值得注意的是,对于 Standard LWE 问题,可以使用 T 变换[28],在花费不超过 $O(n^2)$ 的实例下,将其转化为 Normal Form LWE 问题。从而使用 Bai 等的格基去更高效地求解。2 种不同的格基优劣势见表 1。

表 1 Primal 攻击中 2 种不同格基的优劣势

Table 1 Advantages and disadvantages of two different lattice-based primal atlacks

	[BG14][31]	[AFG14][30]	[BG14][31]加上[ACPS09][32]中 T 变换
无需求逆	√	×	×
s 任意分布	×	√	√

文献[30]中有结论:对于普通形式的 LWE(除 Normal Form LWE 以外),Kannan 嵌入和 Dual 嵌入的效果基本无差别。

第二种对格基约减算法成功求解 u-SVP 的估计由文献[33]给出,核心思想就是 BKZ 算法在求出目标向量时有:$\pi_{m-\beta}(\boldsymbol{e})\approx\sigma\sqrt{\beta}\leqslant\|\boldsymbol{b}^*_{m-\beta}\|$,再由高斯启发式和几何级数假设可得

$$\sqrt{\frac{\beta}{m+1}}\|(\boldsymbol{e}\,|\,1)\|\approx\sqrt{\beta\sigma}\leqslant\delta_0^{2\beta-(m+1)}vol(L(\boldsymbol{A}))^{\frac{1}{m+1}}。$$

该估计与 BKZ 类算法的实际实验结果更为贴切。目前,NIST 第三轮中基于 LWE 困难假设方案的比特安全强度均采用这种由文献[33]中提出的保守估计(采用最为保守 Core-SVP 模型,详情见表 2)[34-35]。该估计通过选择试遍实例数 m 的可取值(对标准型 LWE 来说 $m\in(0,n\log q)$,对 Normal Form 和 Binary 型 LWE 来说 $m\in(0,2n)$),求出当前 m 下使得上述不等式成立的最小 β 值的方法,找到使得攻击开销最小的(m,β)对,最后用 Core-SVP 模型估计出求解方案的安全参数设置下 LWE 问题的计算开销作为方案的比特安全强度。

表 2 SVP 求解器复杂度估计模型

Table 2 Complexity estimation model of SVP solver

模型		复杂度
Core-SVP	Core-SVP	$2^{0.292\beta+16.4}$
	Core-SVP(min space)	$2^{0.3366\beta+12.31}$
	Q-Core-SVP	$2^{0.265\beta+16.4}$
β-SVP	β-SVP	$\beta 2^{0.292\beta+16.4}$
	Q-β-SVP	$\beta 2^{0.265\beta+16.4}$
$8m$-SVP	$8m$-β-SVP	$8m\cdot 2^{0.292\beta+16.4}$
	Q-$8m$-SVP	$8m\cdot 2^{0.265\beta+16.4}$

注:每种评估模型中的第一个子类,例如 Core-SVP 表示经典情形下的模型;第二个子类,例如 Q-Core-SVP 表示加入 Grove 量子搜索后的情形

使用文献[33]中的估计对上述 2 种攻击策略与使用 08 估计[10]得到的结论一致。

3.2.4 对 Binary 型 LWE 问题攻击的加速技术

另外,针对 Binary LWE 问题,其秘密向量 $\boldsymbol{s}$ 的每个分量都取自{0,1}或{-1,0,1},使用全同态加密中的模转换技术和文献[31]中的 Rescale 技术,可以让最终构造格基的 gap 增大,降低攻击算法的复杂度。此外,利用组合攻击也可以一定程度上降低求解秘密向量分布稀疏的 LWE 问题复杂度[36-38]。

(1)模转换技术

模转换是全同态加密中为加速模乘运算速度提出的技术。因为固定其他参数、缩小模数,将使得计算效率提升。但注意到模转换技术虽然减少了模数,但也将引入一部分额外的噪声。对于求解 LWE 问题来说,噪声的增大会导致 LWE 问题的求解越发困难。因此将大模数 q 换成小模数 p 的模转换操作需要合适地取 p 的值。

特别地,一个参数为 n,q,α 的 LWE 实例($\boldsymbol{a}$,b)$\in\mathbb{Z}_q^n\times\mathbb{Z}_q$,利用模转换技术可以被转化为一个参数为 n,$p\approx\sqrt{\pi n/6}\cdot(\sigma_s/\alpha)$,$\sqrt{2}\alpha$($\sigma_s$ 是 $\boldsymbol{s}$ 中元

素的标准差)的形如($\lceil p/q\rceil \cdot \boldsymbol{a}\rceil$,$\lceil (p/q)\cdot \boldsymbol{b}\rceil$) $\in \mathbb{Z}_p^n \times \mathbb{Z}_p$的新的 LWE 实例,其中,$\lceil (p/q)\cdot \boldsymbol{b}\rceil$计算过程如下:

$$\left\lceil \frac{p}{q}\cdot \boldsymbol{b}\right\rceil = \left\lceil \frac{p}{q}(\langle \boldsymbol{a},\boldsymbol{s}\rangle + \boldsymbol{e})\right\rceil =$$

$$\left\lceil \left\langle \left\lceil \frac{p}{q}\cdot \boldsymbol{a}\right\rceil ,\boldsymbol{s}\right\rangle_p + \left\langle \frac{p}{q}\cdot \boldsymbol{a} - \left\lceil \frac{p}{q}\cdot \boldsymbol{a}\right\rceil ,\boldsymbol{s}\right\rangle_p + \frac{p}{q}\cdot \boldsymbol{e}\right\rceil =$$

$$\left\langle \left\lceil \frac{p}{q}\cdot \boldsymbol{a}\right\rceil ,\boldsymbol{s}\right\rangle_p + e''\frac{p}{q}\cdot \boldsymbol{e} + e'。$$

其中,$u \in \mathbb{Z}$,$e' \in [-0.5,0.5]$,$\langle x,y\rangle_p$ 表示在模 p 的意义下对向量 x 和 y 求内积。

如上文所述,当模数 p 减小时,算法的时间复杂度会降低;但是当 p 取得过小时,又会造成噪音规模的增加,导致解决这个实例的困难度增加,因此,p 的取值需要平衡这两者之间的矛盾。最优的 p 值的选择可以使得$\langle (p/q)\cdot \boldsymbol{a} - \lceil (\mathrm{p/q})\cdot \boldsymbol{a}\rceil, \boldsymbol{s}\rangle \approx |(\mathrm{p/q})\cdot \boldsymbol{e}|$,即在模转换后的错误规模大致放缩为原来的 p/q,据此,可以计算出 p 的最优取值约等于 $\sqrt{\pi n/6}\cdot(\sigma_S/\alpha)$。

模转换技术可用于原始攻击和对偶攻击中。

(2)Rescale 技术

Rescale 技术由 Bai 等[31] 在 2014 年提出,这种技术可以被应用在 Dual 嵌入中,并将其称为 BG 嵌入。在 BG 嵌入中,嵌入格被修改为 $L(\boldsymbol{M}) = \{x \in \nu\,\mathbb{Z}^n \times \mathbb{Z}^{m+1} \mid x\cdot(\boldsymbol{A}\,|\,\boldsymbol{I}_m\,|\,-\boldsymbol{b})^{\mathrm{T}} = 0 \bmod q\}$ 形式,基 $\boldsymbol{M} = \begin{pmatrix} v\boldsymbol{I}_n & -\boldsymbol{A}^{\mathrm{T}} & 0 \\ 0 & q\boldsymbol{I}_m & 0 \\ 0 & \boldsymbol{b} & 1 \end{pmatrix}$,且有$(\boldsymbol{s}\,|\,*\,|1)\cdot\boldsymbol{M} = (v\boldsymbol{s}\,|\,\boldsymbol{e}\,|1)$。在普通的 Dual 嵌入情况下时,当 $v=1$,由于 $\boldsymbol{s}$ 每个分量很小或很稀疏时,向量$(\boldsymbol{s}|\boldsymbol{e}|1)$的各个部分的模长是不一样长的。要使$\|\boldsymbol{s}\|/\sqrt{n} \approx \|\boldsymbol{e}\|/\sqrt{m} = \sigma$ 才能将目标向量的各个部分变成均衡的,即需要选择合适的 v 值使得 $\|v\boldsymbol{s}|\boldsymbol{e}|1\| \approx \sigma\sqrt{n+m}$,可以使得调整后的向量($\|v\boldsymbol{s}|\boldsymbol{e}|1\|$)就是 $L(\boldsymbol{M})$ 中的一个最短向量,且利用格基约化算法找到它变得更加高效[31]。因为实际实验表明格基约减算法更加倾向于输出每个分量值都比较均衡的短向量[34]。以 $\boldsymbol{s} \xleftarrow{\$} \{-1,0,1\}^n$ 为例,有 $\|v\boldsymbol{s}\|^2 = 2v^2n/3$,因此,令 $v = \sqrt{3/2}\sigma$,可以算出 $\|v\boldsymbol{s}\|^2 = \sigma\sqrt{n}$,那么有$\|(v\boldsymbol{s}|\boldsymbol{e}|1)\| \approx \sigma\sqrt{n+m}$。

(3)组合攻击

在秘密向量 s 取值稀疏的情况下,如$\{0,1\}$或$\{-1,0,1\}$采用组合攻击[36-39],即给一个秘密 $\boldsymbol{s}$ 最多 $w<n$ 个非零元素,猜测 $\boldsymbol{s}$ 中 k 个元素均是 0,从而减少所求解格的维数。对 $\boldsymbol{s}$ 的每一种猜测,使用格基约减算法看是否能够求出目标向量$(\boldsymbol{s}|\boldsymbol{e}|1)$。这种组合攻击的总开销为 $\min_k\{C(n-k)/P_k\}$,其中,$C(n)$是在 n 维格上执行格基约减算法的开销,而 P_k 是猜测正确的概率[31]。

4　格基约减算法

经过上述分析可知,无论是求解判定性 LWE 问题还是求解搜索型 LWE 问题,都是通过构造某个格基将 LWE 问题转化为构造格上的 u-SVP 问题进行求解,或是对构造格的格基进行约减得到质量足够“好”的格基,从而能够成功解码出目标向量。所以高效的格基约减算法是求解 LWE 问题的主要方法。作为求解近似 SVP 和近似 CVP 的近似算法,目前主要的格基约减算法有 LLL、基于枚举的 BKZ 类以及基于筛法的 General Sieve Kernel(G6K),其中,基于近些年筛法发展最新结果的 G6K 是当前最快的格基约减算法。在简单地回顾经典的 LLL 和 BKZ 类之后,将着重介绍 G6K 算法。介绍格基约减算法之前,首先将 BKZ 类以及 G6K 算法所基于的 SVP 求解算法做一下梳理。

4.1　SVP 求解算法

4.1.1　枚举法

枚举算法的优势是算法开始后是确定性的,且只使用多项式大小的空间。其缺点就是时间复杂度是超指数的 $O(2^{n\log(n)})$。此外,多年来已经开发了一些优化和启发式方法,使得枚举目前在实践中对于较小的维度最具吸引力。枚举是一种通过系统地枚举空间中某个给定半径的超球(通常是 n 维平行六面体或椭球体)内所有格点来解决任意格上最短向量问题(SVP)和最近向量问题(CVP)的标准技术。人们对格枚举技术的兴趣主要是由于它们的低内存需求(关于格的维数 n 是线性的),以及在中等低维数上优异的实际性能。枚举算法的运行时间在很大程度上取决于输入格基的质量。因此,使用格基约减算法对输入格进行适当的预处理是枚举法的重要组成部分。最早

考虑执行预处理后再做枚举的是文献[40-41],其中,文献[40]使用的是LLL算法作为预处理,经过LLL算法处理后,再执行枚举求解出最短向量的开销是$2^{O(n^2)}$。而Kannan[41]提出了一种更加复杂的预处理方法,对枚举算法进行低维递归调用,将(总)渐近运行时间减少到$n^{O(n)}$。文献[42]则给出了Kannan算法在最好情况下时间复杂度的最优分析。文献[42]表明Kannan算法求解SVP的时间为$n^{O(n/(2e))}$。然而,由于其指数时间的预处理开销,对实际可以执行枚举算法的维数n来说,Kannan算法并不比渐近复杂度更劣势的方法(如文献[40]基于多项式时间格基约减算法的变体)快。2015年,Micciancio等[43]引入了Kannan算法的变体,该算法的渐进时间是同Kannan算法一样的$n^{O(n)}$,但文献[43]所需预处理开销更少,使得该算法在中低维的枚举方法中有着很强的竞争力。在实际使用时,枚举算法的有效实现需要大量使用浮点运算,但由于舍入误差和精度问题,浮点运算可能相当棘手。有关使用浮点运算的SVP枚举的详细分析,请参阅文献[44]。

2021年,文献[45-48]将近几年枚举的新技术,如文献[45-47]中放松裁剪枚举算法条件的技术,以及文献[48]中扩展预处理技术相结合,提出了渐进复杂度更低的枚举算法,其渐进复杂度为$n^{O(n/(8))}$。此外,他们将这个新的枚举算法应用到BKZ(2.0)算法中,减少了基于筛法实现的BKZ算法[13]与基于枚举实现的BKZ算法之间实际性能的差距[49]。

4.1.2 筛法

筛法是通过在固定半径的球壳上堆积格点来求解最短向量问题。2001年,Ajtai等[50]提出的随机筛法被认为是最早且最著名的指数空间复杂度算法之一,它的时间复杂度和空间复杂度均为$2^{O(n)}$。其后,Regev[51]、Nguyen等[52]、Micciancio等[53],先后给出了AKS筛法更准确的时间和空间复杂度估计,分别为$2^{3.4n+o(n)}$和$2^{1.97n+o(n)}$。Micciancio等[53]提出了一种确定性的算法,ListSieve该算法将时间和空间复杂度分别改进到了$2^{3.199n+o(n)}$和$2^{1.325n+o(n)}$,该方法结合生日攻击可以在$2^{2.465n+o(n)}$的时间内找到最短向量。2015年,Aggarwal等[54]基于离散高斯采样提出了新的随机算法,将时间复杂度进一步改进到了$2^{n+o(n)}$,此外,在启发式假设下,筛法的复杂度上界可以进一步降低。2008年,Nguyer等[52]基于AKS筛法提出了第一个随机启发式筛法,分别达到了$2^{0.415n+o(n)}$和$2^{0.2075n+o(n)}$的时间和空间复杂度,其他启发式筛法的改进工作还包括二重筛法和三重筛法[55-58]等,当前已知最高效的经典计算机模型下的筛法是由Anja等[59]提出的,该算法实现了时间和空间复杂度的平衡,2者均见文献[59-60]。在量子计算模型下,Larhoven等[60]通过Grover算法将上述筛法的时间复杂度进一步改进为$2^{0.265n+o(n)}$[60]。Voroni细胞算法是由Daniele等[61]于2010年提出的一种新型确定性算法,该算法利用格点的Voronoi细胞结构,将求解相关向量问题的方法用于求解最近向量问题的预处理,其时间和空间复杂度约为$2^{2n+o(n)}$和$2^{n+o(n)}$。

4.2 对经典格基约减算法的梳理

4.2.1 LLL算法

LLL算法是一个多项式时间算法[62],可视为是在二维拉格朗日/高斯算法在高维空间上的扩展。理论上,LLL算法的Hermite因子根可以达到$\delta_0=(4/3)^{(n-1)/4n}\approx 1.075$,而实际运行LLL算法的输出结果要好过其理论结果,LLL算法的Hermite因子根实际可达1.0219[10]。在小维数格上,LLL算法甚至能求出最短向量。

LLL算法的细节描述:

定义5 (LLL约减基)一组基$\boldsymbol{B}=\{\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n\}\in\mathbb{R}^n$称为$L(\boldsymbol{B})$的$\delta-LLL$既约基,如果以下条件成立:

1. $\forall 1\leqslant i\leqslant n, j<i, |\mu_{i,j}|\leqslant 0.5$;

2. $\forall 1\leqslant i\leqslant n, \delta\|\boldsymbol{b}_i^*\|^2\leqslant\|\mu_{i+1,i}\boldsymbol{b}_i^*+\boldsymbol{b}_{i+1}^*\|^2$。

其中,$\boldsymbol{b}_1^*,\boldsymbol{b}_2^*,\cdots,\boldsymbol{b}_n^*$是$\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n$的*Gram-Schmidt*正交化向量,$\mu_{i,j}=\langle\boldsymbol{b}_i,\boldsymbol{b}_j^*\rangle/\langle\boldsymbol{b}_j^*,\boldsymbol{b}_j^*\rangle$。由于$\boldsymbol{b}_i^*$和$\boldsymbol{b}_{i+1}^*$正交,定义中的第二个条件也可以写为

$$\|\boldsymbol{b}_{i+1}^*\|^2\geqslant(\delta-\mu_{i+1,i}^2)\|\boldsymbol{b}_i^*\|^2。$$

又由第一个条件可知,$\|\boldsymbol{b}_{i+1}^*\|^2\geqslant\left(\delta-\frac{1}{4}\right)\|\boldsymbol{b}_i^*\|^2$,即$\boldsymbol{b}_{i+1}^*$的长度不会比$\boldsymbol{b}_i^*$的长度短很多。

下面的定理说明了,这样的基是存在的并且能够在多项式时间内找到。

定理1 任给格基$\boldsymbol{B}=\{\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n\}\in\mathbb{Q}^n$,任意$1/4<y<1$,则LLL算法在多项式时间内输出

$L(\boldsymbol{B})$的一组 y-LLL 约减基。

当 $y=1$ 时,现在还没有从理论上证明 LLL 算法能在多项式时间内结束;但是当 $y=1/4+(3/4)^{n/(n-1)}$时,LLL 算法能在多项式时间内结束,较精确的估计:定义 $D>\|\boldsymbol{b}_i\|$,对所有的 i 从 1 到 n 取值,LLL 算法的时间复杂度为 $O(n^{5+\varepsilon}\log^{2+\varepsilon}D)$[62],后续的研究分别给出了更精确的时间复杂度估计 $O(n^{5+\varepsilon}\log_2 D+n^{4+\varepsilon}\log_2^2 D)$[63] 以及 $n^3\log_2^2 D$(基于启发式假设)[14]。

LLL 算法的实现细节:

输入:格基 $\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n\in\mathbb{Z}^n$ 及参数 y;

输出:$L(\boldsymbol{B})$的 y-LLL 约减基.

1. 计算相应的 *Gram-schmidt* 正交化向量 $\boldsymbol{b}_1^*,\boldsymbol{b}_2^*,\cdots,\boldsymbol{b}_n^*$

2. 循环 *for*$(i=2,2\leqslant i\leqslant n,i++)$

3. 循环 *for*$(j=i-1,1\leqslant j\leqslant i-1,j--)$

4. $\boldsymbol{b}_i=\boldsymbol{b}_i-c_{i,j}\boldsymbol{b}_j$,其中,$c_{i,j}=\dfrac{\langle\boldsymbol{b}_i,\boldsymbol{b}_j^*\rangle}{\langle\boldsymbol{b}_j^*,\boldsymbol{b}_j^*\rangle}$

5. *end for*

6. *end for*

7. *if* $\exists i$, s. t $\delta\|\boldsymbol{b}_i^*\|>\|\mu_{i+1,i}\boldsymbol{b}_i^*+\boldsymbol{b}_{i+1}^*\|$ then

8. $\boldsymbol{b}_i\leftrightarrow\boldsymbol{b}_{i+1}$

9. goto 1

10. *end if*

11. 输出 $\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n$

接下来,给出 y-LLL 约减基的一些性质,为了表述简单,令 $y=3/4$,对于其他的 $1/4<y<1$,有完全类似的结果:

性质:令 $\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n$ 为格 $L\in\mathbb{Q}^n$的一组 LLL 既约基,$\boldsymbol{b}_1^*,\boldsymbol{b}_2^*,\cdots,\boldsymbol{b}_n^*$ 为其相应的 *Gram-Schmidt* 正交化,则

1. $\|\boldsymbol{b}_1\|\leqslant 2^{(n-1)/2}\lambda_1(L)$;

2. $\|\boldsymbol{b}_1\|\leqslant 2^{(n-1)/4}\det(L)^{1/n}$;

3. 对任意 $t\leqslant n$ 个线性无关向量 $\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_n\in L$,则对 $1\leqslant j\leqslant t$,有

$$\|\boldsymbol{b}_j\leqslant 2^{(n-1)/2}\max_{1\leqslant i\leqslant t}\|\boldsymbol{x}_i\|。$$

由于行列式是格的一个重要不变量,且可以在多项式时间内求得,故如上性质的第二条给出的 $\boldsymbol{b}_1$ 长度的上界经常被用到。实际上,有更一般的结果:对于格 L 的一组 LLL 约减基,当 $1\leqslant t\leqslant n$ 时,有 $\|\boldsymbol{b}_t\|\leqslant 2^{\frac{n(n-1)}{4(n-t+1)}}\det(L)^{\frac{1}{n-t+1}}$;当 $t=1$ 或者 $1\leqslant t\leqslant n$,但 $\|\boldsymbol{b}_t\|\geqslant 2^{(t-2)/2}$时,有 $\|\boldsymbol{b}_t\|\leqslant 2^{\frac{n+t-2}{4}}\det(L)^{\frac{1}{n-t+1}}$

4.2.2　BKZ 类算法

(1)BKZ 算法

BKZ 算法是由 Schnorr 等[9]应用分块约减的思想改进 LLL 算法而提出的。衡量 BKZ 算法的关键参数是分块大小的 β,β 越大,BKZ 输出约减基的质量越好,输出的最短向量也越短,但同时算法运行时间也越长。

BKZ 算法的每一轮会依次在 $L_{[i;i+\beta]}$上运行枚举算法(其中,$i\in\{1,2,\cdots,n-\beta\}$,当 $i\in\{n-\beta+1,\cdots,n\}$时,则在 $L_{[i;n]}$上运行枚举算法),并将枚举出的最短向量嵌入格基 B,然后用 LLL 算法去除线性相关性,从而逐渐提高格基质量。当某一轮运行结束后,最短向量模长与上一轮输出的结果相比没有改变,则 BKZ 算法终止,并输出当前的最短向量。下面给出 BKZ 算法输出基的质量和算法运行时间的估计。

1)质量估计

对分块大小为 β 的 BKZ 算法的 Hermite 因子为 $\delta_\beta^{\frac{n-1}{\beta-1}}$[14],其中,$\delta_\beta$ 是关于 β 的单调递减函数,并随着 β 增大而逐渐趋向于 1^+。一般对 δ_β 的估计采用 Chen[11] 给出的估计:$\left[\dfrac{\beta}{2\pi e}(\pi\beta)^{\frac{1}{\beta}}\right]^{\frac{1}{2(\beta-1)}}$或是文献[12]中的估计:$\beta^{1/2\beta},2^{1/\beta}$。

2)时间估计

BKZ 算法最耗时的部分就是多项式次调用枚举算法求解 β 维投影子格上最短向量。因此,BKZ 的运行时间可以表示为 $T_{BKZ}=poly(m)T_{svp}(\beta)$,其中 $T_{svp}(\beta)$表示在 β 维格执行一次枚举算法所需的时间[53]。总的来说,BKZ 算法的运行时间关于分块长度 β 是指数级别的,从最初文献[64]中的 $2^{O(\beta^2)}$降到现在的 $2^{O(\beta)}$[53]。而对于 BKZ 类算法所需运行的轮数 ρ,当前尚无准确的估计,最优的上界是指数级的,在实际分析中,常见的估计包括 $\rho=1(core-SVP)$,$\rho=8k$ 和 $\rho=\dfrac{n^2}{k^2}\log_2 n$[6]。所以,对总时间中项 $poly(m)$的准确大小尚无估计。此外,Chen 等[14]还给出了一个用以估计 BKZ2.0 输出基质量的 BKZ2.0 算法模拟运行程序[65]。

对于一组 BKZ 约化基,有如下假设成立:一组 BKZ 约化基 Gram-Schmidt 正交化之后的向量的范数满足 $\|\boldsymbol{b}_i^*\|\geqslant\alpha^{i-1}\|\boldsymbol{b}_1\|(0<\alpha<1)$,这一假设称为几何级数假设(Geometric Series Assumption,

GSA)[66]。由 Hermite 因子的定义可知 $\|\boldsymbol{b}_1\| = \delta_0^m vol(\Lambda)^{\frac{1}{m}}$,另外,显然有 $vol(\Lambda) = \prod_{i=1}^{m} \|\boldsymbol{b}_i^*\|$,据此可计算几何级数假设中的参数 $\alpha = \delta_0^{-\frac{2m}{m-1}} \approx \delta_0^{-2}$。对于 BKZ 约减基,几何级数假设一般认为是成立的。但更精细地,文献[67]证明了几何级数假设仅对 $m-\beta$ 个 Gram-Schmidt 正交化向量成立,对于后面的 β 个 Gram-Schmidt 正交化向量,则满足关系式 $\|\boldsymbol{b}_i^*\| \geqslant (\sqrt{\beta})^{-\log(\beta-1)} \|\boldsymbol{b}_1\|$。可见,随着下标增长,$\|\boldsymbol{b}_i^*\|$ 减小的速度比 GSA 中更快[14]。

(2)BKZ2.0 算法

2011 年,Chen 等[14]对 BKZ 算法进行了 4 点改进,得到 BKZ2.0 算法[65]:

1)提前终止算法:原始 BKZ 算法的运行终止条件是执行完本轮约减得到的格基较之上一轮约减得到的格基没有任何变化,则终止 BKZ 算法。但文献[11]等实验表明,BKZ 算法执行一定的轮数后,继续执行 BKZ 算法对格基质量的改善作用越来越少。但要达到格基不能作进一步改善的条件,仍需执行很多轮 BKZ。因此,BKZ2.0 将算法的终止条件改成格基的质量无明显改善时终止。

2)极度裁剪:如 4.1.1 小节中所描述那样,枚举通过搜索固定某个半径内的 n 维平行六面体或椭球体内所有的格点来解决任意点阵上最短向量问题。Gama 等[68]通过重复多次:随机地裁剪掉搜索空间中绝大部分格点,只枚举剩下的少量格点的极度裁剪枚举,获得了总时间开销更少的枚举算法。

3)强预处理:原始的 BKZ 算法在执行前是使用 LLL 算法做格基的预处理。BKZ2.0 算法采用比输入分块大小更小的 BKZ 算法做 BKZ2.0 算法执行前的预处理。这样虽然使得预处理的开销比原来大,但能获得比 LLL 算法输出基质量更高的预处理结果,如此可有效降低调用枚举算法的计算开销[37]。

4)优化枚举算法的搜索半径:原始 BKZ 算法调用枚举算法求解投影子格上的最短向量时枚举空间的搜索半径都是固定的。但实际上随着枚举的层数逐渐增加,目标向量下一个坐标的可能取值范围是逐渐减少的。BKZ2.0 对枚举每一层的搜索半径进行了估计,并用优化的估计半径代替了原先固定的大半径,减少了枚举算法的计算开销[14]。

其中,2)~4)有 3 个改进的目的都是减少 BKZ 在调用枚举算法时的计算开销。

此外,基于 BKZ2.0 算法,2021 年,Albrecht 等将放松裁剪条件[45-47]以及扩展预处理等技术[48]相结合提出了新的 BKZ 算法,减少了基于筛法实现的 BKZ 算法[13]与基于枚举实现的 BKZ 算法之间实际性能的差距[49]。

(3)Progressive-BKZ 算法

2016 年,Aono 等[45]对 BKZ 算法中分块长度 β 的取值方式进行了更细致的分析,提出了渐近型 BKZ(Progressive-BKZ,P-BKZ)算法。P-BKZ 算法与 BKZ,BKZ2.0 最大的不同就是其分块的大小不是固定的,而是随着 P-BKZ 算法的执行由一个小的初值逐渐增大。P-BKZ 算法通过合适选取的分块大小初值和递增值,使得算法整体的时间复杂度最优。与 BKZ2.0 相比,P-BKZ 算法在解决最高 160 维的 SVP 挑战(SVP challenge)[15]时比 BKZ2.0 最多可以快约 50 倍[45]。

(4)BKZ 算法模拟器

最早的 BKZ 模拟器由 Chen 等[14]提出。此后 Aono 等观察到 BKZ 约化基 Gram-Schmidt 范数曲线相对于 GSA 曲线的“头部凹陷”现象并提出了分析和利用它提升 BKZ 效率的方法,并作出一个新的模拟程序。2017 年,Yu 等[69]进行了大量实验对 BKZ 的实际行为进行了评估,对于 BKZ 约化基 Gram-Schmidt 正交化后范数的分布进行了更为细致的研究,并且通过观察结果更准确地量化了“头部凹陷”现象。2018 年,Bai 等[70]通过考虑随机格中短向量的分布又提出了一个更为精确的 BKZ 模拟程序,很好地模拟并解释了“头部凹陷”现象,并利用该现象构造了 Pressed-BKZ 算法。同时指出该现象随着分块大小的增大而不再明显,在分块大小约为 200 时,该现象几乎完全消失。

(5)Slide-BKZ 算法

除了 LLL 和 BKZ(以及 BKZ2.0)之外,还有一类特殊的格基约化算法,即滑动约化(Slide reduction)[16]和对偶 BKZ(Dual-BKZ)[71],这类算法在理论分析上比 BKZ 算法更优:BKZ 理论上可以输出的最短向量满足 $\|\boldsymbol{b}_1\| \leqslant 2\gamma_k^{\frac{n-1}{2(k-1)}+\frac{3}{2}} vol(L)^{\frac{1}{n}}$,而滑动约化和对偶 BKZ 算法的上界可以减小到 $\gamma_k^{\frac{n-1}{2(k-1)}} vol(L)^{\frac{1}{n}}$,其中,$\gamma_k$ 称为 Hermite 常量,用来表示一个 k 维格 L 的 $(\lambda_1(L)/vol(L)^{\frac{1}{k}})^2$ 的上界。可

以看出,后者的理论值略小于 BKZ,但在实际应用中,对于相同的分块长度 β,BKZ 算法的实际输出基质量要优于这类格基约化算法。

(6)对 BKZ 类算法计算开销的估计

根据文献[72]中的实验结果,一些研究者将时间成本估计式中的 $o(\beta)$ 省略或者替换为常数 16.4。另外,文献[72]还对空间和时间综合考虑,在空间复杂度最低的情况下,成本模型中的参数 $c=0.336\,6$,$o(\beta)$ 替换为常数 12.31。在进行安全性评估时,根据考虑的 SVP oracle 调用次数,可将其分为 3 种模型:Core-SVP 模型[33]、β-SVP 模型[73]和 $8m$-SVP 模型[36]。值得注意的是,文献[13]的结论指出 $8m$-SVP 模型是一个过高的估计(不再保守),但文献[13]的结论仍未打破 Core-SVP 模型的下界,因此,使用 Core-SVP 模型进行估计仍是保守的。

4.3　G6K 算法

4.3.1　G6K 算法的 CPU 实现

General Sieve Kernel(G6K)是 Albrecht 等[13]于 2019 年提出的一种基于筛法的可以执行各类格基约减策略的抽象状态机。这个抽象的状态机可以通过调整不同的约减策略去执行目前已有的各种格基约减算法,如经典的 BKZ 算法。但值得注意的是,经典 BKZ 算法是基于枚举算法作为其在约减过程中求投影子格上最短向量时的 Oracle 调用,而 G6K 是基于筛法执行其格基约减策略的。同时,使用筛法的 G6K 不仅仅像经典 BKZ 类那样将枚举算法作为一个黑盒调用,只输出一个短向量,而是构造了一个筛法的数据库,存储着筛法筛出的大量相当短的向量。

G6K 结合并扩展了近年来筛法的最新优化改进成果,其中主要代表是 Ducas[19]在 2018 年欧密上对子筛法的研究。提出对于求解 n 维格上 SVP,对这个 n 维格的 n—d 维投影子格做筛法,而不是对 n 维完整格做筛法,并利用最近平面算法从投影子格筛出的候选短向量中提升出 n 维完整格上最短向量的策略。将求解 n 维格上 SVP 降低至求解 n—d 维投影子格上 SVP,理论上 $d=O(n/\log n)$时,Ducas 证明子筛法均可成功[19]。实际上自由维数 d 的取值可能比理论估计更大,需要根据实例具体实验才能准确得到。此外,G6k 中还利用的一项技术是 Laarhoven 等[74]在 2018 年 PQC 上提出的渐进筛。其主要思想与 Progressive BKZ 的想法类似,从小的投影子格开始做筛法,然后逐渐增大所筛子格维数,利用投影子格上已筛出的短向量组成的数据库加速加入新格基向量后做筛法的速度。下面给出这 2 个技术的详细描述。

(1)Dimension for free[19]

Ducas 提出的 Dimension for free(Dimforfree)技术。Dimforfree 技术的主要思想是:对于要求解的 n 维格上最短向量问题并不是像之前一样对整个 n 维格直接进行筛法求解,而是只需几次调用在 n—d 维原格投影子格上的筛法,以及利用低维格上的最近平面算法,将投影子格上求解出的短向量提升到 n 维完整格上的最短向量的组合算法。如此,就将最耗时的筛法所筛格的维数从 n 维降至 n—d 维。其中,$d=n/\log n$ 时,Ducas[19]证明该子筛法仍能够成功,尽管 Ducas 这个改进的提升只是亚指数的 $2^{n/\log n}$,但在实际实验中有显著的提升效果。需要指出的是,具体实现时,Ducas 同样没有找到一个很好的对 d 的准确估计,只能通过实际测试的方法从大的 d 值逐渐减少,直到 d 减少到能求出原问题的解。

Ducas 的具体做法是:d 先选一个很大的值,如 $n/4$,然后 d 每次减少 1,直到子筛法最终找到最短向量(d 减小到 $n/\log n$ 时,就达到理论上子筛法成功条件了),取更大的 d 纯粹为了更快求解。

对 n 维格的 n—d 维投影子格 L_d 做筛法:

$$L:=Sieve(L_d)=\{\vec{x}\in L_d\setminus\{0\}\mid\|\vec{x}\|\leqslant\sqrt{4/3}\cdot gh(L_d)\}$$

如此,获得投影子格 L_d 上的很多很短的格向量,用 $\vec{s}_d$ 表示,则有 $\vec{s}_d=\pi_d(\boldsymbol{B}\vec{x})=\boldsymbol{B}_d\vec{x}''$,令 $\boldsymbol{B}=[\boldsymbol{B}'\mid\boldsymbol{B}'']$,$\vec{x}'\in\mathbb{Z}^d$,$\vec{x}''\in\mathbb{Z}^{n-d}$,$\vec{s}_d=\boldsymbol{B}_d\vec{x}''$在原格上其中一个原像为 $\boldsymbol{B}''\vec{x}''$,通过基 $\boldsymbol{B}_d$ 和 $\vec{s}_d$ 可将 $\vec{x}''$求出来。$\vec{s}_d$ 短是 $\boldsymbol{B}''\vec{x}''$在投影子格 L_d 上的某个原像 $\boldsymbol{B}''\vec{x}''$短的必要条件。对 n—d 维投影子格 L_d 做的筛法保证了 $\vec{s}_d$ 足够短。Ducas 则给出了一个较为严格的条件:$gh(L)\leqslant\sqrt{4/3}\cdot gh(L_d)$。接下来只需找到 $L(\boldsymbol{B}')$ 上的格点 $\boldsymbol{B}'\vec{x}'$;使得 $\boldsymbol{B}'\vec{x}'+\boldsymbol{B}''\vec{x}''$最短即可。这显然是一个 d 维格 $L(\boldsymbol{B}')$ 上的 BDD 问题,目标向量是 $-\boldsymbol{B}''\vec{x}''$。如此加上 $\boldsymbol{B}'\vec{x}'$的 $\boldsymbol{B}''\vec{x}''$向量将得到更进一步的约减。Ducas 采用 Babai 最近平面算法来求解这个 BDD 问题,并给出了该投影子格筛加上 Babai 算法的组合算法求解 n 维格上 SVP 问

题的很宽松的成功条件:对所有比 d 小的角标 i:$gh(L) \leqslant \frac{1}{2}\min_{i<n} \| b_i^* \|$。Ducas 指出,用分块大小为 $n/2$ 的 BKZ-β 算法做 n 维子筛法的预处理操作,该 BKZ 算法至多调用 $n/2$ 次维数为 $n/2$ 的筛法作为最短向量问题的求解 Oracle,则 n 维子筛法最终只用对 $n—d=n—n/\log(n)$ 维投影子格做筛法,并在 d 维子格 $L(\boldsymbol{B}')$ 上使用多项式时间的 Babai 最近平面算法,就能求出 n 维格上最短向量问题。如此改进的筛法计算复杂度比对 n 维格的完全筛法少一个亚指数因子:$2^{n/\log(n)}$。

Ducas 利用 Dimforfree 技术构造了一个能求出近似 HKZ 约减基的子筛法$^+$算法,最后,再用输出的近似 HKZ 约减基 $\boldsymbol{V}$ 代替当前格 L 的格基向量,作为新的输出格基 $\boldsymbol{B}$。具体做法就是利用 LLL 算法对[$\boldsymbol{V}$ | $\boldsymbol{B}$]进行约减,并去除其中线性相关的部分。

(2)[LM18] 渐近筛法[74]

[LM18] 渐近筛法的基本思想是首先对输入格基做格基约减的预处理。其次从某个维数小于完整格的维数子格开始做筛法,筛出的结果中实际上已经包含了很多短的格向量,通过这些信息,可以更快地找到完整格上近似最短向量。一个更清晰地解释渐进筛比经典筛法更快的理由是:当筛法输入的格基约减程度越高(格基质量越好),那么在较低维数的子格中,筛法会找到更短的格向量并将其添加到列表 L 中,而列表中有更短的向量,意味着:扩大子格维数后,重新抽样得到的新的格向量可以更容易更快地被已经在列表中的短向量所约减。由于在经典的筛法(直接在高维格上执行筛法)中需要花费更长的时间先找到(近似)较短的格向量。同时经典的筛法不能像渐进筛一样,在这种加速约减中获益(在每一个低维子格中筛出的更短向量都将有助于列表中向量的质量,让下一次抽样出的新向量被更短的向量所约减)。[LM18][74]相信在很大程度上解释了当使用约减程度更高的格基作为渐近筛的输入时,所带来的计算效率上的有效提升。

渐进筛相较于经典筛法的优势在于:更少的迭代次数就能求出 SVP 的解,更少的内存开销,预处理对其求出速度影响更为明显,更容易预测筛法的行为。

介绍完 G6K 算法中使用的主要改进筛法的技术后,再对 G6K 算法本身进行描述。

(3)G6K 算法内部状态、指令和约减策略[13]

所有被 G6K 所考虑的向量都在投影子格 $L_{[l:r]}$ 中。更准确地说,$\boldsymbol{w}=\boldsymbol{B}_{[l:r]}\cdot\boldsymbol{v}\in\mathbb{R}^d$,其中,$\boldsymbol{v}\in\mathbb{Z}^n$。用格点 $\boldsymbol{w}$ 在投影格基 $\boldsymbol{B}_{[l:r]}$ 下的坐标向量 $\boldsymbol{v}$ 来表示格点。这种表示转换的开销为 $O(n^2)$。接下来定义 3 种对向量分量的扩展或是缩减操作:

1)G6K 内部状态表示

-格基 $\boldsymbol{B}\in\mathbb{Z}^{d\times d}$ 在每次使用最近平面算法做提升操作有新的格向量嵌入时需要更新,同时,$\boldsymbol{B}$ 的 Gram-Schmidt 正交基 $\boldsymbol{B}^*$ 也要随着更新。

-位置参数 $0\leqslant k\leqslant l\leqslant r\leqslant d$。$[l:r]$ 表示筛法所筛投影子格所在格基的下标范围,$[k:r]$ 表示提升嵌入格向量时,嵌入格向量可嵌入的格基下标范围。定义 $n=r—l$ 为所筛子格维数。

-用 db 表示对投影子格 $L_{[l:r]}$ 做筛法筛出的 N 个格向量组成的数据库。

-候选嵌入向量 $c_k,\cdots,c_l$,其中,$c_i\in L_{[i:r]}$ 或 $c_i=\perp$。

2)G6K 基本指令

-$\mathrm{Init_B}$:初始化格基。对 G6K 状态机进行初始化,初始格基设置为 $\boldsymbol{B}\in\mathbb{Z}^{d\times d}$。

-$\boldsymbol{Reset}_{k,l,r}$:重置格向量数据库 db。将格向量数据库设置为空集,并设置参数(k, l, r)

-Sieve(S):对选中投影子格做筛法,并将筛出的短向量通过最近平面算法嵌入到合适位置。在执行对 $L_{[l:r]}$ 的筛法过程中,将筛出的短向量嵌入到 $L_{[k:r]}$ 上的合适位置。如果嵌入能改善格基的质量,则称 on-the-fly lifting。

-(ER,SL,EL):右扩展,左缩减,左扩展。向右或向左增加或减少数据库中每一个向量的分量。

-Insert(I):根据评分函数选择最优的嵌入位置 c_i,其中,$k\leqslant i\leqslant l$。同时,每次嵌入需要更新格基。如果没有合适的嵌入位置,则继续运行 SL 以确保所筛法能正常终止。

-$\boldsymbol{Resize}_N$:将数据库大小调整至 N。当需要缩减数据库大小时,将数据库中最长的向量移除;当增大数据库大小时,则抽样新的格向量加入到数据库中。

3)G6K 约减策略表示

作为可以执行不同格基约减策略的抽象状态机。首先将 G6K 的基本指令按要执行约减的策略

组合成称为 Pump 的指令集合:

$$Pump_{k,f,\beta,s}:Reset_{k,k+\beta,k+\beta},\overbrace{(EL,S)^{\beta-f}}^{\text{pump-up}},\overbrace{(I,S)^{\beta-f}}^{\text{pump-down}},$$

其中,$0\leqslant k\leqslant k+\beta\leqslant d$,$0\leqslant f\leqslant\beta$,$s\in\{0,1\}$ 控制在 Pump-down 阶段是否执行筛法。$Reset_{k,k+\beta,k+\beta}$表示对数据库的重置,其中,$k$ 确定此次 Pump 中数据库格向量可嵌入位置的下标下界。Pump-up 阶段利用子筛法和渐进筛的思想,每次迭代对数据库中全体格向量执行一次左扩展(EL)操作:将数据库中全体格向量利用 Babai 算法从 $L_{[i;r]}$ 提升到 $L_{[i-1;r]}$后(其中,$i\in[l+1,r]$),都要对整个数据库执行一次筛法(Sieve(S)操作)。经过 $\beta-f$ 次迭代以后,获得存有大量短的格向量数据库 db,随后进入 Pump-down 阶段。Pump-down 阶段中据评分函数[31,75]将 $L_{[l;r]}$ 格中筛出的短的格向量提升嵌入到 $L_{[k;r]}$ 中评分最高的位置,得到新格基。嵌入操作利用 Dimforfree 技术和嵌入评分函数,实现了对格基的质量提升。

最后,G6K 利用一系列 Pump 指令集合的组合,刻画不同的格基约减策略:

$$WorkOut_{k,\beta,f,f^+,s}:Pump_{k,\beta-f^+,\beta,s},\ Pump_{k,\beta-2f^+,\beta,s},\ Pump_{k,\beta-3f^+,\beta,s},\cdots,\ Pump_{k,f,\beta,s},$$

其中,f^+ 表示每次迭代 Pump 时,所筛投影子格维数的递增量。整个 G6K 算法实现的整体框架可以用图 2 表示。

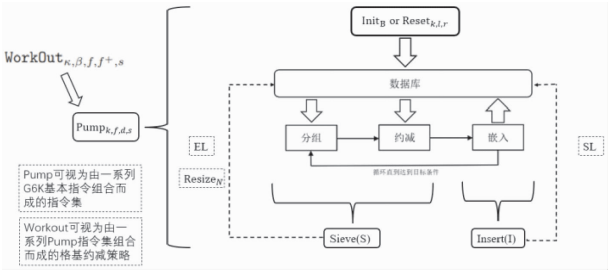

图 2　G6K 算法整体框架图

Fig. 2　Overal frame diagram of G6K algorithm

图 2 中的分组步骤是 G6K 所基于的启发式筛法在执行格向量约减前的一个分组操作。不同启发式筛法的最主要区别就在于它们对向量的不同分组方法,从而导致它们的时间复杂度和找到相对接近向量对的性能不同。以经典的 BGJ 筛法为例介绍分组过程:利用 NNS 技术[75],首先将数据库中众多格向量用一个个超锥体分成多个不同的组,其中同一个组中的格向量与指示超锥体方向的中心向量夹角小于某个预设值。分组后考虑同一组中格向量对之间是否存在约减,而不是考虑整个数据库中全部可能的向量对组合是否存在约减。如此可有效减少筛法的计算复杂度。详情见 BGJ 筛[75]以及 BDGL 筛[59]。

G6K 算法默认执行的格基约减策略是:首先将 r 设置为 d(d 是输入格基的维数),l 设置为 $d-n_0$,其中,n_0 表示首次 Pump 的投影子格维数上界(默认取值是 40)。Pump-up 阶段中每次 Pump 的初始子格维数为 30,经过 Pump-up 完成对 $L_{[l;r]}$ 投影子格的渐进筛。随后进入 Pump-down 阶段,根据评分函数将 $L_{[l;r]}$ 格中筛出的短向量嵌入到 $L_{[k;r]}$ 中的合适位置并更新格基。下一次 Pump 时,采用子筛法的思想,使用 Lift 先将整个数据库中格向量提升到 $L_{[l-f^+;r]}$ 中,即令 $l=l-f^+$,将下一次 Pump 所筛投影子格维数上界增大 f^+(f^+ 默认值是 3),再次迭代。直到某次 Pump 时,嵌入的格向量模长小于预设的目标向量模长,就终止程序并返回最短格向量,或是当前 Pump 的子格维数达到 $\beta-f$ 也终止算法。

因此,通过调整 k,β,f,f^+,s 可以构造各种约减策略不同的格基约减算法。例如,经典的 BKZ 算法的约减策略可以写成:

$$NaiveTour_\beta:WorkOut_{0,\beta},\ WorkOut_{1,\beta+1},\cdots\ WorkOut_{d-\beta,d},\cdots,\ WorkOut_{d-1,d}$$

[LM18][74]中滑动窗口(Sliding-window)策略可以表示成:

$$SlidingWindowTour_\beta:$$

$$\mathrm{Reset}_{0,0,0},(ER,S)^\beta,(I_l,S,ER,S)^{d-\beta},(I_l,S)^\beta.$$

此外,G6K 在 Pump 的初始阶段默认采用 Gauss 筛法以确保初始时不遗漏太多格向量。随着 Pump 的进行,对维数大于 40 的投影子格,G6K 默认使用 hk3 筛法。随着筛法技术的进步,可以将更快或是空间复杂度更小的筛法替换掉 G6K 当前所使用的筛法。

4.3.2　G6K 的 GPU 实现

2021 年,Ducas 等[76]利用 NVIDIA GPU Tensor 核心的高效矩阵计算能力极大加速了 G6K 算法执行筛法时对向量分组和向量约减的效率。同时,所引入的 Dual-hash 技术在 G6K 算法执行 EL 操作时过滤掉了很多并不值得提升的向量,有效减少了提升操作的开销。

提升了此前只使用 CPU 实现的 G6K 算法的

计算效率,同时极大地降低了能耗(对180维随机格上SVP挑战,GPU实现的能耗不到CPU实现能耗的4%)。他们在Darmstadt的SVP挑战中创下了求解180维随机格上求最短向量的纪录,此前最好的纪录由CPU实现的G6K算法创下,只能求到155维。

5 总 结

本文整理并比较了目前求解LWE问题的分析策略,并对目前主要的格基约减算法的思想和所使用的技术进行了梳理。对于不同类型的LWE问题,虽然分析策略有所不同,但大都可以用格基约减算法完成求解。对于现有的格基约减算法,仍有很多改进的地方,例如,如何通过优化底层筛法去降低G6K算法的GPU实现所需的内存开销,利用G6K所定义的抽象状态机找到约减效果更好的格基约减策略,等等。此外,如何更准确地预测格基约减算法输出基的质量,从而能够对基于LWE问题的困难性构造密码方案的安全性更准确地估计也是笔者下一步工作的重点。尤其是对诸如G6K等新的格基约减算法输出基质量的预测,目前尚无很好的估计,笔者将在未来对这一问题做进一步的研究。

参考文献:

[1] Ajtai M, Dwork C. A public-key cryptosystem with worst-case/average-case equivalence[C]//Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing. New York: ACM, 1997:284-293.

[2] Regev O. On lattices, learning with errors, random linear codes, and cryptography[C]//STOC'05: Proceedings of the thirty-seventh Annual ACM Symposium on Theory of Computing. New York:ACM, 2005:84-93.

[3] Gentry C, Peikert C, Vaikuntanathan V. Trapdoors for hard lattices and new cryptographic constructions[C]//STOC'08: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. New York:ACM, 2008:197-206.

[4] Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE[C]//2011 IEEE 52nd Annual Symposium on Foundations of Computer Science. Piscataway: IEEE,2011:97-106.

[5] Agrawal S, Boneh D, Boyen X. Efficient lattice (H)IBE in the standard model[C]//EUROCRYPT. 2010. Lecture Notes in Computer Science. Berlin:Springer, 2010:553-572.

[6] Albrecht M R, Player R, Scott S. On the concrete hardness of learning with errors[J]. Journal of Mathematical Cryptology, 2015, 9(3): 169-203.

[7] Kannan R. Minkowski's convex body theorem and integer programming[J]. Mathematics of Operations Research, 1987, 12(3):415-440.

[8] Albrecht M R, Ducas L. Lattice attacks on NTRU and LWE: A history of refinements[J]. IACR Cryptol. ePrint Arch, 2021(10):799. https://eprint.iacr.org/2021/799.

[9] Schnorr C P, Euchner M. Lattice basis reduction: Improved practical algorithms and solving subset sum problems[C]//Fundamentals of Computation Theory. FCT 1991. Lecture Notes in Computer Science. Berlin: Springer, 1991:68-85.

[10] Gama N, Nguyen P Q. Predicting lattice reduction[C]//Advances in Cryptology-EUROCRYPT of Lecture Notes in Computer Science. Berlin: Springer, 2008:31-51.

[11] Chen Y M. Réduction de réseau et sécurité concrète du chiffremeni comlètement homomorphe[D]. Paris: University Paris Diderot (Paris 7), 2013.

[12] Stehle D. An overview of lattice reduction algorithms[C]//Proceedings of the 2017 ACM on Internation Symposium on Symbolic and Algebraic Computation, 2017:11-12. doi:10.1145/3087604.3087665.

[13] Albrecht M R, Ducas L, Herold G, et al. The general sieve kernel and new records in lattice reduction[C]//Advances in Cryptology-EUROCRYPT 2019. Lecture Notes in Computer Science. Cham: Springer, 2019, doi:10.1007/978-3-030-17656-3_25.

[14] Chen Y, Nguyen P Q. BKZ 2.0: Better lattice security estimates[C]//Advances in Cryptology-ASIACRYPT 2011. Lecture Notes in Computer Science. Berlin:Springer, 2011, doi:10.1007/978-3-642-25385-0_1.

[15] TU Darmstadt Department of Computer Science. TU darmstadt lattice challenge[EB/OL]. (2021-11-16)[2021-11-28]. http://www.latticechallenge.org/svp-chall-enge/.

[16] Gama N, Nguyen P Q. Finding short lattice vectors within mordell's inequality[C]//STOC'08: Proceedings of the Fortieth

Annual ACM Symposium on Theory of Computing, 2008:207-216.

[17] Zhang X, Zheng Z X, Wang X Y. A detailed analysis of primal attack and its variants[J]. Science China Information Sciences, 2022, 65(3): 132301. https://doi.org/10.1007/s11432-020-2958-9.

[18] Schnorr C. Block reduced lattice bases and successive minima[J]. Combinatorics, Probability and Computing, 1994, 3(4): 507-522.

[19] Ducas L. Shortest vector from lattice sieving: A few dimensions for free[C]//Nielsen J, Rijmen V (eds). Advances in Cryptology-Eurocrypt 2018. Lecture Notes in Computer Science. Cham:Springer, 2018,doi:10.1007/978-3-319-78381-9_5.

[20] Tobias Hamann. The BKZ simulation algorithm[J]. Fachbereich Informatik Fachgebiet Theoretische Informatik-Kryptographie und Computeralgebra. 2013,doi:10.1.1.380.4063.

[21] Blum A, Kalai A, Wasserman H. Noise: Tolerant learning, the parity problem, and the statistical query model[J]. Journal of the ACM,2003,50(4):506-519.

[22] Arora S, Ge R. New algorithms for learning in presence of errors[C]//Automata, Languages and Programming. ICALP 2011. Lecture Notes in Computer Science. Berlin:Springer,2011, doi:10.1007/978-3-642-22006-7_34.

[23] Albrecht M R, Faugère J C, Fitzpatrick R, et al. Lazy modulus switching for the BKW Algorithm on LWE[C]//Public-Key Cryptography-PKC 2014. Lecture Notes in Computer Science. Berlin:Springer, 2014:429-445.

[24] Guo Q, Johansson T, Stankovski P. Coded-BKW: Solving LWE using lattice codes[C]//Advances in Cryptology-CRYPTO 2015. Lecture Notes in Computer Science. Berlin:Springer, 2015, doi:10.1007/978-3-662-47989-6_2.

[25] Kirchner P, Fouque P A. An improved BKW algorithm for LWE with applications to cryptography and lattices[C]//Advances in Cryptology-CRYPTO 2015. Lecture Notes in Computer Science. Berlin:Springer, 2015, doi:10.1007/978-3-662-47989-6_3.

[26] Guo Q, Johansson T, Märtensson E, et al. Coded-BKW with sieving[C]//Advances in Cryptology-Asiacrypt 2017. Lecture Notes in Computer Science. Cham:Springer, 2017:323-346.

[27] Albrecht M R, Cid C, Faugère J C, et al. Algebraic algorithms for LWE problems[J]. ACM Communications in Computer Algebra, 2015, 49(2):62.

[28] Lindner R, Peikert C. Better key sizes (and attacks) for LWE-based encryption[C]//Topics in Cryptology-CT-RSA 2011. Lecture Notes in Computer Science. Berlin:Springer, 2011:319-339.

[29] Babai L. On Lov'asz' lattice reduction and the nearest lattice point problem (shortenedversion)[J]. Symposium of Theoretical Aspects of Computer Science(STACS'86), 1986,6:13-20.

[30] Albrecht M R, Fitzpatrick R, Göpfert F. On the efficacy of solving LWE by reduction to unique-SVP[C]//Information Security and Cryptology-ICISC 2013. Lecture Notes in Computer Science. Cham:Springer,2014:293-310.

[31] Bai S, Galbraith S D. Lattice decoding attacks on binary LWE[C]//Susilo W, Mu Y. (eds) Information Security and Privacy. ACISP 2014: Proceedings of the 19th Australasian Conference on Information Security and Privacy. Lecture Notes in Computer Science. Cham: Springer, 2014,doi:10.1007/978-3-319-08344-5_21.

[32] Applebaum B, Cash D, Peikert C,et al. Fast cryptographic primitives and circular-secure encryption based on hard learning problems[C]//Advances in Cryptology-CRYPTO 2009. Lecture Notes in Computer Science. Berlin:Springer, 2009:595-618.

[33] Alkim E, Ducas L, Pöppelmann T, et al. Post quantum key exchange-A new hope[C]//Usenix Security Symposium (USENIX Security'16). Austin, TX: USENIX Association, 2016:327-343.

[34] Albrecht M R, Göpfert F, Virdia F, et al. Revisiting the expected cost of solving uSVP and applications to LWE[C]//Takagi T, Peyrin T. (eds) Advances in Cryptology-ASIACRYPT 2017. ASIACRYPT 2017. Lecture Notes in Computer Science, 2017, 10624. Cham: Springer. doi:10.1007/978-3-319-70694-8_11.

[35] Chen L L, Moody D, Liu Y K. Post-quantum cryptography PQC round 3 submissions, round 3 finalists: Public-key encryption and key-establishment algorithms; round 3 finalists:Digital signature algorithms[EB/OL]. (2021-12-02)[2021-12-21]. https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions.

[36] Albrecht M R. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL[C]//Coron J S, Nielsen J (eds) Advances in Cryptology-EUROCRYPT 2017. Lecture Notes in Computer Science. Cham:Springer, 2017:103-129.

[37] Howgrave-Graham N. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU[C]//Advances in Cryptology-

Crypto 2007. Lecture Notes in Computer Science. Berlin:Springer,2007,doi:10.1007/978-3-540-74143-5_9.

[38] Teruya T, Kashiwabara K, Hanaoka G. Fast lattice basis reduction suitable for massive parallelization and its application to the shortest vector problem[C]//Public-Key Cryptography-PKC 2018. Lecture Notes in Computer Science. Cham:Springer, 2018:437-460.

[39] Guo Q, Johansson T. Faster dual lattice attacks for solving LWE with applications to CRYSTALS[C]//Tibouchi M, Wang H. (eds) Advances in Cryptology-ASIACRYPT 2021. ASIACRYPT 2021. Lecture Notes in Computer Science, 2021, 13093. Cham: Springer. doi:10.1007/978-3-030-92068-5_2.

[40] Fincke, Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis[J]. Mathematics of Computation,1985,44(170):463-471.

[41] Kannan. Minkowski's convex body theorem and integer programming[J]. Mathematics of Operations, Research, 1983,12(3):415-442.

[42] Hanrot, Stehle. Improved analysis of kannan's shortest lattice vector algorithm[C]//Advances in Cryptology-CRYPTO 2007. CRYPTO 2007. Lecture Notes in Computer Science. Berlin:Springer, 2007, doi:10.1007/978-3-540-74143-5_10.

[43] Micciancio D, Walter M. Fast lattice point enumeration with minimal overhead[J]. Crypotology and Information Security Series,2014,2014:569.

[44] Pujol, Stehlé. Rigorous and efficient short lattice vectors enumeration[C]//Conference: Advances in Cryptology-Asiacrypt 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security. Belin: Springer, 2008:5350.

[45] Aono Y, Wang Y, Hayashi T, et al. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator[C]//Advances in Cryptology-Eurocrypt 2016. EUROCRYPT 2016. Lecture Notes in Computer Science. Berlin:Springer,2016, doi:10.1007/978-3-662-49890-3_30.

[46] Aggarwal D, Li J, Nguyen P Q, et al. Slide reduction, revisited—filling the gaps in SVP approximation[C]//Advances in Cryptology-Crypto 2020. Lecture Notes in Computer Science. Cham:Springer, 2020, doi:10.1007/978-3-030-56880-1_10.

[47] Li J W, Nguyen P Q. A complete analysis of the BKZ lattice reduction algorithm[J/OL]. (2020-10-07)[2021-09-28]. https://eprint.iacr.org/2020/1237.pdf, 2020.

[48] Albrecht M R, Bai S, Fouque P A, et al. Faster enumeration-based lattice reduction: Root hermite factor $k^{1/(2k)}$ time $k^{k/8+o(k)}$[C]//Advances in Cryptology-Crypto 2020. Lecture Notes in Computer Science. Cham:Springer, 2020, doi:10.1007/978-3-030-56880-1_7.

[49] Albrecht M R, Bai S, Li J, et al. Lattice reduction with approximate enumeration oracles[C]//Advances in Cryptology-Crypto 2021. Lecture Notes in Computer Science. Cham:Springer,2021, 12826,doi:10.1007/978-3-030-84245-1_25.

[50] Ajtai M, Kumar R, Sivakumar D. A sieve algorithm for the shortest lattice vector problem[C]//STOC'01: Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing. 2001: 601-610.

[51] Regev O. Lecture notes on lattices in computer science[EB/OL]. (2011-09-26)[2021-09-28]. 2004. http://www.cs.tau.ac.il/odedr/teaching/lattices fall 2004/index.html.

[52] Nguyen P Q, Vidick T. Sieve algorithms for the shortest vector problem are practical[J]. Journal of Mathematical Cryptology, 2008. 2(2): 181-207.

[53] Micciancio D, Voulgaris P. Faster exponential time algorithms for the shortest vector problem[J]. Electronic Colloquium on Computational Complexity, 2009,10: 1468-1480.

[54] Aggarwal D, Dadush D, Regev, et al. Solving the shortest vector problem in 2 time using discrete gaussian sampling[C]//STOC'15: Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing. New York:Association for Computing Machinery, 2015: 733-742.

[55] Laarhoven T. Sieving for shortest vectors in lattices using angular locality-sensitive hashing[C]//Advances in Cryptology-Crypto 2015. Lecture Notes in Computer Science. Berlin: Springer, 2015, doi:10.1007/978-3-662-47989-6_1.

[56] Thijs Laarhoven, Benne de Weger. Faster sieving for shortest lattice vectors using spherical locality-sensitive hashing[C]//Progress in Cryptology-Latincrypt 2015. Lecture Notes in Computer Science. Cham:Springer, 2015,doi:10.1007/978-3-319-22174-8_6.

[57] Wang X Y, Liu M J, Tian C L, et al. Improved nguyen-vidick heuistic sieve algorithm for shortest vector problem[C]//ASIACCS'11: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. New York:

ACM, 2011: 1-9.

[58] Zhang F, Pan Y B, Hu G R. A three-level sieve algorithm for the shortest vector problem[C]//Selected areas in cryptography-SAC 2013. Lecture Notes in Computer Science. Berlin:Springer, 2014, doi:10.1007/978-3-662-43414-7_2.

[59] Anja B, Léo D, Nicolas G, et al. New directions in nearest neighbor searching with applications to lattice sieving[C]//Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms. SIAM, 2016:10-24.

[60] Laarhoven T, Michele M, Joop van de Pol. Solving the shortest vector problem in lattices faster using quantum search[C]// Post-Quantum Cryptography. PQCrypto 2013. Lecture Notes in Computer Science. Berlin:Springer, 2013, doi:10.1007/978-3-642-38616-9_6.

[61] Daniele M, Voulgaris P. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations[C]//STOC'10: Proceedings of the Forty-second ACM Symposium on Theory of Computing. New York:ACM, 2010: 351-358.

[62] Lenstra A K, Lenstra H W, Lovász L. Factoring polynomials with rational coefficients[J]. Mathematische Annalen, 1982, 261(6): 515-534.

[63] Nguen P Q, Stehle D. Floating-point LLL revisited[C]//Advances in Cryptology-Eurocrypt 2005. Lecture Notes in Computer Science. Berlin:Springer, 2005,doi:10.1007/11426639_13.

[64] Schnorr C P, Euchner M. Lattice basis reduction: Improved practical algorithms and solving subset sum problems[C]// Fundamentals of Computation Theory. FCT 1991. Lecture Notes in Computer Science. Berlin:Springer, 1991, doi:10.1007/3-540-54458-5_51.

[65] Chen Y, Nguyen P Q. BKZ 2.0: Better lattice security estimates[C]//Advances in Cryptology-Asiacrypt 2011. Lecture Notes in Computer Science. Berlin:Springer, 2011,doi:10.1007/978-3-642-25385-0_1.

[66] Schnorr C P. Lattice reduction by random sampling and birthday methods[C]//STACS 2003. STACS 2003. Lecture Notes in Computer Science. Berlin:Springer, 2003,doi:10.1007/3-540-36494-3_14.

[67] Hanrot G, Stehlé D. Improved analysis of kannan's shortest lattice vector algorithm[C]//Menezes A. (eds) Advances in Cryptology-CRYPTO 2007. CRYPTO 2007. Lecture Notes in Computer Science, 2007, 4622. Berlin: Springer,2007, doi:10.1007/978-3-540-74143-5_10.

[68] Gama N, Nguyen P Q, Regev O. Lattice enumeration using extreme pruning[C]//Gilbert H (eds.) Advances in Cryptology-Eurocrypt 2010. Lecture Notes in Computer Science. Berlin:Springer, 2010,doi:10.1007/978-3-642-13190-5_13.

[69] Yu Y, Ducas L. Second order statistical behavior of LLL and BKZ[C]//Selected Areas in Cryptography-SAC 2017. Lecture Notes in Computer Science. Cham:Springer, 2018,doi:10.1007/978-3-319-72565-9_1.

[70] Bai S, Stehlé D, Wen W Q. Measuring, simulating and exploiting the head concavity phenomenon in BKZ[C]//Advances in Cryptology-Asiacrypt 2018. Asiacrypt 2018. Lecture Notes in Computer Science. Cham:Springer, 2018, doi:10.1007/978-3-030-03326-2_13.

[71] Micciancio D, Walter M. Practical, Predictable Lattice Basis Reduction[C]//Advances in Cryptology-Eurocrypt 2016. Lecture Notes in Computer Science. Berlin:Springer, 2016,doi:10.1007/978-3-662-49890-3_31.

[72] Laarhoven T. Sieving for shortest vectors in lattices using angular locality-sensitive hashing[C]//Advances in Cryptology-Crypto 2015. Crypto 2015. Lecture Notes in Computer Science. Berlin:pringer, 2015,doi:10.1007/978-3-662-47989-6_1.

[73] Bos J, Costello C, Ducas L, et al, Frodo: Take off the ring! practical, quantum-secure key exchange from LWE[C]// CCS'16: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2016:1006-1018.

[74] Laarhoven T, Mariano A. Progressive lattice sieving[C]//Post-Quantum Cryptography. PQCrypto 2018. Lecture Notes in Computer Science. Cham:Springer, 2018,doi:10.1007/978-3-319-79063-3_14.

[75] Becker A, Gama N, Joux A. Speeding-up lattice sieveing without increasing the memory, using sub-quadratic nearest neighbor search[J]. IACR Cryptol. ePrint Arch, 2015, Report 2015/522. https://eprint.iacr.org/2015/522.pdf.

[76] Ducas L, Stevens M, van Woerden W. Advanced lattice sieving on GPUs, with tensor cores[C]//Advances in Cryptology-Eurocrypt 2021. Eurocrypt 2021. Lecture Notes in Computer Science. Cham:Springer, 2021, doi:10.1007/978-3-030-77886-6_9.

【责任编辑: 陈　钢】