Computer Security Symposium 2015
21 - 23 October 2015

近似 GCD 問題のパラメータ解析

高安 敦 †　　國廣 昇 †

† 東京大学

あらまし 近似 GCD 問題は、素因数分解に関連する解くことが困難であると予想される問題であり、近年その困難性に基づいて整数上の完全準同型暗号や多重線形写像が構成されている。それゆえ、近似 GCD 問題を解く計算時間を解析することは、これらの暗号方式の効率を見積もる上で重要な問題である。本稿では、格子簡約を利用した近似 GCD 問題の解法を解析する。そして、近似 GCD 問題を解くための格子に特化した LLL 簡約アルゴリズムの高速化を提案する。これにより、現在設定されている近似 GCD 問題のパラメータは、予想されていたよりも安全ではないことを示す。

Parameter Selections for Approximate GCD Problems

Atsushi Takayasu†　　Noboru Kunihiro†

†The University of Tokyo.

Abstract Approximate GCD problem is expected to be hard to solve and relates to factorization. Recently, fully homomorphic encryptions and multilinear maps over the integers have been constructed based on the hardness of approximate GCD. Hence, the analysis for the computational complexity of the problem is important to analyze the practicality of fully homomorphic encryptions and multilinear maps. In this paper, we analyze the lattice based method to solve the problem. In particular, we propose faster LLL reduction algorithm on specific lattices to solve approximate GCD problem. As a result, we show that recently proposed parameters are less secure than suggested.

1　導入

近似 GCD 問題. Gentry [Gen09] がイデアル格子上で完全準同型暗号 (fully homomorphic encryption, FHE) を初めて提案し，その後いくつかの構成法が見つかった．その一つは Dijk et al. [DGHV10] による整数上での構成で，様々な改良 [CMNT11, CNT12, CCK+13, CLT14, CS15, KN15] も提案され，広く研究されている．また，Garg ら [GGH13] がイデアル格子上で多重線形写像 (multilinear map, MMP) の候補を初めて構成した後にも，整数上での構成法が提案された [CLT13, CLT15]．特に，整数上の MMP に関しては致命的な攻撃 [CHL+14] も見つかっており，FHE や MMP はまだ理論的な研究対象としての側面が強いが，将来の実用化を視野に入れ，実装上どの程度の効率を達成しえるのかを研究することは意義深い．そのためには，どのようなパラメータ設定のもとで適切な安全性を達成できるかを調べる必要があり，それは FHE や MMP の安全性と密接な関わりを持つ近似 GCD 問題 (approximate GCD problem) の困難性を研究することを意味する．

近似 GCD 問題は Howgrave-Graham [How01] によって初めて導入され，簡単に言えば，秘密

整数 p の 1 つの倍数と多項式個のノイズ付き倍数が与えられたときに p を求める問題である．これまで近似 GCD 問題の困難性を解析する多くの結果が知られている [DGHV10, CMNT11, CN12, CH12, LS14]．その中には LLL アルゴリズムなどの格子簡約アルゴリズムを利用するものがあるが，FHE や MMP で用いるパラメータに対しては非常に大きな入力と次元の基底行列を簡約する必要があり，実装上その安全性を破るほど高速には動かないことが報告されている [CN12]．よって，この攻撃はより LLL アルゴリズムを高速化するような改良が報告されない限り有効ではない．

Rounding LLL. PKC 2014 で Bi ら [BCF+14] は，特定の基底行列に対して LLL アルゴリズムを高速化する方法を提案した．具体的には，彼らの提案した *Rounding LLL* は入力となる基底行列が下三角で全ての対角成分の大きさが近ければ素朴な LLL アルゴリズムを多項式オーダーで高速化する．暗号の安全性評価の文脈では，このような基底行列を持つ例として Coppersmith の法付き方程式を解く手法 [Cop96] が挙げられる．

しかし，Rounding LLL が有効になるような基底行列の性質は非常に限定的である．前述のように，Coppersmith の手法で扱う行列に対して有効だが，これは Coppersmith が提案した法が既知の 1 変数法付き方程式に関する場合だけであり，より強力な場合，法が未知，または，多変数法付き方程式に対しては非常に小さな定数倍しか高速化ができない．そして，格子簡約を用いて近似 GCD 問題を解くときには，法が未知でかつ多変数法付き方程式を解く必要があり，Rounding LLL を使っても大幅な高速化が見込めない．Bi ら [BCF+14] は，彼らの論文中でこのような場合を含むような拡張を重要な open problem としていた．

成果. 本稿で我々は，Bi ら [BCF+14] とは異なる特定の基底行列に対する LLL アルゴリズムの高速化を提案する．我々の手法が対象とする基底は Bi らの Rounding LLL とかなり制約が異なるが，我々の手法は Bi らのものと結合可能であり，真に広い基底に対して LLL を高速化したと言える．我々の手法が対称とする基底は，下三角行列どころか正方行列である必要もなく，第 1 行の入力が他の行の要素より十分大きいときに多項式オーダーで LLL 簡約の高速化を達成する．この制約は一見不自然に思うかも知れないが，近似 GCD 問題を解くための基底はこの制約を満たす．具体的には，入力となる基底行列 $\boldsymbol{B} \in \mathbb{Z}^{n\times m}$ の第 j 行の要素の最大入力長を α_j とすると，$\alpha_1 > n \cdot \max_{j\in[2,m]} \alpha_j$ のとき我々の手法は有効で，L^2 アルゴリズムを用いたとき $\Theta\left((\alpha_1/n)^{2/3}\right)$ 倍の高速化を実現し，$O\left(n^{17/3}\alpha_1^{4/3}\right)$ 時間で終了する．

2 準備

$n' < n$ なる整数 n と n' に対し，$[n]$ を集合 $1, 2, \ldots, n$ とし，$[n', n]$ を集合 $n', n'+1, \ldots, n$ とする．行列 $\boldsymbol{B} \in \mathbb{Z}^{n\times m}$ の第 i 行第 j 列の要素を b_{ij} とする．$\boldsymbol{b}_i \in \mathbb{Z}^m$ と $\vec{b}_j \in \mathbb{Z}^n$ をそれぞれ，第 i 行ベクトル，第 j 列ベクトルとする．$\|\boldsymbol{a}\|$ をベクトル $\boldsymbol{a}$ のユークリッドノルムとし，$\|\boldsymbol{A}\|$ を行列 $\boldsymbol{A}$ の行ベクトルの最大ユークリッドノルムとする．2 つのベクトル $\boldsymbol{b}_i$ と $\boldsymbol{b}_j$ の内積を $\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle$ と書く．2 つの行列 $\boldsymbol{A}$ と $\boldsymbol{B}$ の横と縦の連結をそれぞれ $(\boldsymbol{A}|\boldsymbol{B})$，$(\boldsymbol{A}\|\boldsymbol{B})$ とする．$1 \le m' < m$ なる m' に対し，$\boldsymbol{B}_{[m',m]}$ を行列 B の第 m' 列から第 m 列までからなる部分行列とする，つまり，$\boldsymbol{B}_{[m',m]} = (\vec{b}_{m'}|\cdots|\vec{b}_m)$ である．$\boldsymbol{0}_n$ を n 次元のゼロベクトル，$\boldsymbol{O}_{n,m}$ を $n \times m$ ゼロ行列とする．$\boldsymbol{I}_n$ を $n \times n$ 単位行列とする．

ベクトル $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^m$ は全て線形独立で，これらのベクトルを基底に張られる格子を $L(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) = \{\sum_{j=1}^n c_j \boldsymbol{b}_j | c_j \in \mathbb{Z} \text{ for all } j\}$ とする．また，$\boldsymbol{B} = (\boldsymbol{b}_1\| \cdots \|\boldsymbol{b}_n)$ を格子の基底の行列表現とし，同じ格子を $L(\boldsymbol{B}) = \{\boldsymbol{c}\boldsymbol{B} | \boldsymbol{c} \in \mathbb{Z}^n\}$ と書く．基底 $\boldsymbol{B}$ で張られる格子の体積 $\mathrm{vol}(L(\boldsymbol{B}))$ を領域 $\mathcal{P}(B) := \{\boldsymbol{c}B : \boldsymbol{c} \in \mathbb{R}^n, 0 \le c_j < 1, \text{for all } j = 1, \ldots, n\}$ の体積で定義し，$\mathrm{vol}(L(\boldsymbol{B})) = \sqrt{\det(\boldsymbol{B}\boldsymbol{B}^T)}$ で計算できる．

$\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ は線形独立な m 次元整数ベクトルとし，それらの Grahm-Schmidt 直行化 $\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*$ を，$\boldsymbol{b}_1^* = \boldsymbol{b}_1$ とし，残りを $\boldsymbol{b}_i^* = \boldsymbol{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \boldsymbol{b}_j^*$ で計算する．ただし，$\mu_{i,j} = \frac{\langle \boldsymbol{b}_i, \boldsymbol{b}_j^* \rangle}{\langle \boldsymbol{b}_j, \boldsymbol{b}_j^* \rangle}$ とする．格

子の体積は $\mathrm{vol}(L(\boldsymbol{B})) = \prod_{i=1}^{n} \|\boldsymbol{b}_i^*\|$ でも計算できる．

次の 2 つの条件を満たすとき，格子の基底 $\boldsymbol{B}$ は（パラメータ δ に関して）LLL 簡約されているという; (1) 全ての $1 \le j < i \le n$ に対し，$|\mu_{i,j}| \le \frac{1}{2}$，(2) $i \in [n-1]$ に対し，$\delta\|\boldsymbol{b}_i^*\|^2 \le \|\mu_{i+1,i} \cdot \boldsymbol{b}_i^* + \boldsymbol{b}_{i+1}^*\|^2$．LLL 簡約された基底のノルムは次のような性質を持つ．

補題 1 ([May03]) 基底 $\boldsymbol{B} \in \mathbb{Z}^{n \times m}$ が *LLL* 簡約ならば，全ての $i \in [n]$ に対して

$$\|\boldsymbol{b}_i\| \le 2^{n(n-1)/(4(n+1-i))} \cdot (\det(\boldsymbol{B}))^{1/(n+1-i)}$$

が成りたつ．

この結果は厳密に証明されているが，LLL アルゴリズムは実装上には理論的に保証される最悪時よりもずっと小さなノルムのベクトルを出力することが知られている．つまり，上記の最悪時のノルムを持つような行列は非常に特殊な場合でしかありえない．本稿では，我々は次の仮定に基づき評価する．証明は省略するが，行列の Grahm-Schmidt ベクトルのノルムが i に対して単調減少ならばこの仮定は成りたつ．この条件は少し強いが，ランダムな入力に LLL アルゴリズムを適用したとき Grahm-Schmidt ベクトルのノルムは減少する傾向にあり，May の厳密な証明による見積もりより現実的であると考える．なお，この仮定の妥当性は実験的に正当性を確かめる．

仮定 1 *LLL* 簡約の基底 $\boldsymbol{B} \in \mathbb{Z}^{n \times m}$ は，ある定数 $c = O(1)$ に対して $\|\boldsymbol{B}\| \le c^n \cdot (\det(\boldsymbol{B}))^{1/n}$ を満たす．

$1/4 < \delta < 1$ のとき，LLL アルゴリズム [LLL82] は多項式時間で LLL 簡約な基底を出力する．b を LLL アルゴリズムの入力となる行列 $B \in \mathbb{Z}^{n \times m}$ の最大入力長とすると，LLL アルゴリズムは $O(n^5 m b^3)$ で終了する．それ以降様々な高速化が提案されており，実装上最も速いのは Nguyen と Stehlé の L^2 アルゴリズム [NS09] で，その計算量は $O(n^4 m(n+b)b)$ である．

整数上の FHE や MMP の論文の記法に従い，γ, η と ρ をそれぞれ公開要素，秘密要素，ノイズのビット長とする．λ を安全性パラメータとするとき，基本的なパラメータ選択は $\gamma = O(\lambda^5), \eta = O(\eta^2), \rho = O(\lambda)$ である．η ビットの秘密要素を p に対して，公開要素 $a_0, a_1, \ldots, a_k$ を次のように計算する; ノイズ無し要素を $a_0 = p \cdot q_0$ ただし $q_0 \hookleftarrow U[0, 2^\gamma/p)$，ノイズ付き要素 $a_1, \ldots, a_k$ を全ての $i \in [k]$ に対して $a_j = p \cdot q_i + r_i$ ただし $q_i \hookleftarrow U[0, 2^\gamma/a_0), r_i \hookleftarrow U(-2^\rho, 2^\rho)$ とする．ただし，$a < b$ なる整数 a と b に対して $U(a,b)$ を区間 (a,b) の整数の一様分布である．近似 GCD 問題は，上記のように定義される公開要素 $a_0, a_1, \ldots, a_k$ を与えられたときに秘密要素 p を計算する問題である．

LLL アルゴリズムを用いて，近似 GCD 問題を解く方法を簡単にまとめる．公開要素を並べた列ベクトルを $\vec{a} = (a_0, a_1, \ldots, a_k)$ とし，$(k+1) \times (k+1)$ 下三角行列 $\boldsymbol{B} = (\vec{a} | (\boldsymbol{0}_k \| 2^\rho \cdot \boldsymbol{I}_k))$ とする．近似 GCD 問題のノイズからなるベクトルを $\boldsymbol{u} = (1, -r_1/2^\rho, \ldots, -r_k/2^\rho)$ とすると，行列 $\boldsymbol{B}$ の全ての列ベクトル $\boldsymbol{b}_i$ とベクトル $\boldsymbol{u}$ の内積は p を法として 0 になる，$\langle \boldsymbol{b}_i, \boldsymbol{u} \rangle = 0 \pmod{p}$．そして，行列 $\boldsymbol{B}$ を基底に張られる格子の全てのベクトルがこの性質を満たし，$\langle \boldsymbol{b}'_i, \boldsymbol{u} \rangle \le \|\boldsymbol{b}'_i\| \cdot \|\boldsymbol{u}\| \le \sqrt{k+1} \cdot \|\boldsymbol{b}'_i\| < p$ を満たすほど小さい格子上のベクトルを見つけることができれば，$\langle \boldsymbol{b}'_i, \boldsymbol{u} \rangle = 0$ が法 p を外して整数上で成り立つため，連立方程式を解くことでノイズ $r_1, \ldots, r_k$ を求めることができる．$\mathrm{vol}(L(\boldsymbol{B})) \le 2^{\gamma + k \cdot \rho}$ より，LLL アルゴリズムを用いて $\|\boldsymbol{b}'_i\| \le c^{k+1} \cdot 2^{\frac{1}{k+1}\gamma + \frac{k}{k+1}\rho}$ なるベクトルを多項式時間で求めることができる．よって，$c^{k+1} \cdot 2^{\frac{1}{k+1}\gamma + \frac{k}{k+1}\rho} < p/\sqrt{k+1}$ なる k が存在するときには，LLL アルゴリズムを用いて近似 GCD 問題を解くことができる．一連の計算時間の大部分は LLL アルゴリズムが占めており，入力長 γ と格子の次元 $k+1$ に依存する．

3 Rounding LLL

先行研究． この章で，Bi ら [BCF+14] の提案した Rounding LLL を紹介し，その素朴な改良を示す．前章で見たように，近似 GCD 問題を解くための基底行列は下三角で第 1 列のみ入力

が大きい行列であったが，この章ではより一般的な下三角行列 $\boldsymbol{B} \in \mathbb{Z}^{n\times n}$ に LLL アルゴリズムを適用することを考える．具体的には，全ての $j \in [n]$ に対して対角成分が $\log |b_{j,j}| < \alpha_j$ を満たし，各列の要素の絶対値は対角成分よりも小さいとする; 全ての $1 \le j \le i \le n$ に対し，$|b_{i,j}| \le |b_{j,j}|$．この行列 $\boldsymbol{B}$ に素朴に L^2 アルゴリズムを適用すると，最大入力長が $\max_{j\in[n]} \alpha_j$ で行列の次元が n なので，$O(n^5 \cdot (\max_{j\in[n]} \alpha_j)^2)$ 時間かかり，$\|\boldsymbol{B}'\| \le c^n \cdot 2^{\sum_{i=1}^n \alpha_i/n}$ なる行列 $\boldsymbol{B}'$ を出力する．

Bi ら [BCF+14] の Rounding LLL を紹介する．Rounding LLL の高速化は，素朴に LLL を適用するのではなく，入力となる基底行列の各要素の上位ビットのみに LLL 簡約を行なうことによるものである．具体的には，基底行列 $\boldsymbol{B}$ を入力としたとき，パラメータ μ の Rounding LLL は次の 3 ステップからなる．(1) 各要素を $t_{i,j} = \left\lfloor b_{i,j} / \left(2^{\min_{j\in[1,n]} \alpha_j - \mu}\right) \right\rfloor$ とする行列 $\boldsymbol{T} = (t_{i,j}) \in \mathbb{Z}^{n\times n}$ を生成する．(2) L^2 アルゴリズムを行列 $\boldsymbol{T}$ に適用し，その LLL 簡約行列 $\boldsymbol{S}$ を計算する．(3) $\boldsymbol{B}' = \boldsymbol{S} \cdot \boldsymbol{T}^{-1} \cdot \boldsymbol{B}$ を出力する．そして，Rounding LLL は以下の性質を満たす．

定理 1 ([BCF+14]) 定数 $\mu > 0$ をパラメータ，行列 $B \in \mathbb{Z}^{n\times n}$ を入力とするとき，上記の *Rounding LLL* アルゴリズムは L^2 を用いると時間 $O(n^5 \cdot (\max_{j\in[1,n]} \alpha_j - \min_{j\in[1,n]} \alpha_j + \mu)^2)$ で終了する．また，出力 $\boldsymbol{B}'$ は

$$\|\boldsymbol{B}'\| \le \left(1 + n \cdot \left(\frac{3}{2} + \frac{1}{2(2^\mu - 1)}\right)^{n-1} \cdot 2^{-\mu}\right) \cdot c^n \cdot 2^{\sum_{i=1}^n \alpha_i/n}$$

を満たす．

定理 1 の証明の概観を述べる．まず，出力のノルムについて述べる．行列 $\boldsymbol{T}$ の定義から，その行列式は $\det(\boldsymbol{T}) \le \prod_{j=1}^n 2^{\alpha_j - (\min_{j\in[n]} \alpha_j - \mu)} = 2^{\sum_{j=1}^n \alpha_j - n(\min_{j\in[n]} \alpha_j - \mu)}$ となり，LLL 簡約された $\boldsymbol{S}$ は $\|\boldsymbol{S}\| = c^n \cdot 2^{\sum_{j=1}^n \alpha_j/n - (\min_{j\in[n]} \alpha_j - \mu)}$ を満たす．行列 $\bar{\boldsymbol{B}} = \boldsymbol{B} - 2^{\min_{j\in[1,n]} \alpha_j - \mu} \cdot \boldsymbol{T}$ は全ての $i, j \in [n]^2$ に対して $|\bar{b}_{i,j}| < 2^{\min_{j\in[1,n]} \alpha_j - \mu}$ を満たし，行列 $\boldsymbol{B}$ の下位ビットとなっていることから，$\|\boldsymbol{B}'\| = \|\boldsymbol{S} \cdot \boldsymbol{T}^{-1} \cdot \boldsymbol{B}\| = \|\boldsymbol{S}\| \cdot \|\boldsymbol{T}^{-1} \cdot (2^{\min_{j\in[1,n]} \alpha_j - \mu} \cdot \boldsymbol{T} + \bar{\boldsymbol{B}})\| = \|\boldsymbol{S}\| \cdot \|2^{\min_{j\in[1,n]} \alpha_j - \mu} \cdot \boldsymbol{I}_n + \boldsymbol{T}^{-1} \cdot \bar{\boldsymbol{B}}\|$ となる．詳細 (Lemma 4, [BCF+14]) は省略するが，$\|\boldsymbol{T}^{-1}\| \le \sqrt{n} \cdot \left(\frac{3}{2} + \frac{1}{2(2^\mu - 1)}\right)^{n-1} \cdot 2^{-\mu}$ が成りたつため，定理 1 のノルムが得られる．

次に，素朴な LLL 簡約からの改良の核となる計算時間に関してだが，計算時間の大部分はステップ (2) の L^2 アルゴリズムによる．このとき，行列 $\boldsymbol{T}$ は入力 $\boldsymbol{B}$ と比べると各成分を $2^{\min_{j\in[1,n]} \alpha_j - \mu}$ で割っているため入力が小さくなっており，素朴に LLL アルゴリズムを適用するよりも計算量が減る．定義から最大入力長が $\max_{j\in[1,n]} \alpha_j - \min_{j\in[1,n]} \alpha_j + \mu$ の $n \times n$ 行列であり，L^2 アルゴリズムを用いると定理 1 の計算量になる．Coppersmith の定理 [Cop96] で用いる基底行列は $\max_{j\in[1,n]} \alpha_j - \min_{j\in[1,n]} \alpha_j$ の値が小さくなるため，Rounding LLL を用いることで大幅に計算量の削減に繋がる．しかし，近似 GCD 問題を解く基底行列では素朴に LLL 簡約をするときとの差は非常に小さく，小さな定数倍の改良にしかならない．詳しくは [BCF+14] の Appendix G を参照されたし．

また，Rounding LLL を使うときの注意点として，ステップ (3) で行列 $\boldsymbol{T}^{-1}$ を求めている点である．この処理のため行列 $\boldsymbol{T}$ は正則行列でなければならず，この制約を満たすために Rounding LLL が適用できる入力には下三角行列という条件が必要である．

素朴な拡張． Rounding LLL は，特定の基底行列を入力とするときの LLL 簡約を高速化する手法であるが，近似 GCD 問題を解くための基底に対してはその効果は薄い．より具体的には，$\max_{j\in[1,n]} \alpha_j - \min_{j\in[1,n]} \alpha_j$ の値がほぼ $\max_{j\in[1,n]} \alpha_j$ と変わらないときには素朴に L^2 アルゴリズムを適用したときとの計算量が変わらない．この欠点は，出力 $\boldsymbol{B}'$ がほぼ LLL 簡約されるように，アルゴリズムのステップ (1) で入力基底 $\boldsymbol{B}$ の各要素を $2^{\min_{j\in[1,n]} \alpha_j - \mu}$ で割って丸めた行列 $\boldsymbol{T}$ を作り，L^2 アルゴリズムを適用することに依る．

もし，入力基底をより大きな値で割ることで行列 $\boldsymbol{T}$ を作ると，計算時間はずっと短くなるが出力される行列は LLL 簡約されたものと比べる

とノルムが非常に大きくなる．この単純なアプローチで Rounding LLL よりも高速に，かつ，LLL 簡約された行列を出力することができることを示す．つまり，前述の処理を事前処理と考え，そこで出力された行列にもう一度 L^2 アルゴリズムを適用すれば，出力する行列は必ず LLL 簡約されている．それゆえ，この L^2 アルゴリズムを 2 度適用しても，1 度しか用いない素朴な L^2 アルゴリズムや Rounding LLL よりも高速化されていれば良い．

基底行列 $\boldsymbol{B}$ とパラメータ μ を入力とし，4 ステップからなる次の拡張 Rounding LLL を考える．(1) 各要素を $t_{i,j} = \lceil b_{i,j}/2^{\alpha_1/2-\mu} \rfloor$ とする行列 $\boldsymbol{T} = (t_{i,j})$ be a $n \times n$ を生成する．(2) L^2 アルゴリズムを行列 $\boldsymbol{T}$ に適用し，その LLL 簡約行列 $\boldsymbol{S}$ を計算する．(3) $\boldsymbol{B'} = \boldsymbol{S} \cdot \boldsymbol{T}^{-1} \cdot \boldsymbol{B}$ を計算する．(4) 行列 $\boldsymbol{B'}$ を入力とし L^2 アルゴリズムを適用し，その結果 $\boldsymbol{B''}$ を出力する．ここからは，近似 GCD 問題を解くための基底を念頭に，$\alpha_1 > n \cdot \max_{j\in[2,n]} \alpha_j$ のときのみを考える．つまり，入力基底は 1 行目のみ非常に大きい要素を持ち，他の行の要素は小さい．このとき，証明は省略するが拡張 Rounding LLL は次の性質を持つ．

定理 2 (拡張 Rounding LLL) 拡張 *Rounding LLL* の出力は *LLL* 簡約されており，$\alpha_1 > n \cdot \max_{j\in[2,n]} \alpha_j$ のとき，ステップ *(2)* と *(4)* の L^2 アルゴリズムの最大入力長は $\alpha_1/2 + O(n)$ を満たす．

L^2 アルゴリズムの入力長がほぼ $\alpha_1/2$ であると仮定すると，L^2 アルゴリズムの計算量が入力長の 2 乗のオーダーであることから，1 回の計算で約 4 倍高速化されており，2 度適用したとしても合計約 2 倍高速化されており，これは素朴な Rounding LLL よりも速い．よって，対角成分の大きさに偏りがあるときには拡張 Rounding LLL はより有効である．

直感的には Rounding LLL は $2^{\min_{j\in[n]} \alpha_j}$ を基準により上位のビットの情報のみを用いて基底を簡約している．そして，拡張 Rounding LLL は $2^{\alpha_1/2}$ を基準にそれを行なっており，つまり，ステップ (2) の簡約で上位半分のビットの情報を用いて簡約をし，ステップ (4) で残りの情報を用いて簡約をしている．単純に考えると，拡張 Rounding LLL の方針のまま簡約を行なう情報をより分割することでより改良を見込めそうだが，それは不可能である．つまり，L^2 アルゴリズムを τ 回適用することで，各入力長がほぼ α_1/τ になれば τ 倍の高速化が可能になりそうだが，ステップ (3) の行列 $\boldsymbol{B'}$ はもはや下三角ではないため，Rounding LLL が下三角行列でないと適用できないという事実と同様，拡張 Rounding LLL は $\tau = 2$ のときにしか適用できない．

4 提案手法

この章で，我々は新たな入力基底に対する LLL の高速化を提案する．

定理 3 (提案手法) 全ての $i \in [n]$ と $j \in [m]$ に対して $\log |b_{i,j}| \leq \alpha_j$ を満たす基底行列 $\boldsymbol{B} \in \mathbb{Z}^{n\times m}$ を入力とし，$\alpha_1 = \Omega(n \cdot \max_{j\in[2,m]} \alpha_j)$ を満たすとき，$\alpha_1 < n^4$ ならば L^2 アルゴリズムを用いることでこの *LLL* 簡約を $O(n^{17/3}\alpha_1^{4/3})$ 時間で行うことができる．

提案手法は，先行研究や前章の拡張 Rounding LLL と違い，入力の基底行列が下三角である必要がなく，入力が大きく偏っているときに多項式オーダーでの高速化が可能である．素朴な L^2 アルゴリズムの計算量が $O(n^5\alpha_1^2)$ なので，提案手法は $(\alpha_1/n)^{2/3}$ 倍高速化している．$\alpha_1 < n^4$ という条件は限定的なものではなく，より大きな α_1 に対してはパラメータを変えることで n^2 倍の高速化が可能になるが，我々の知る限りではこのように入力が大きな行列が暗号の安全性解析の文脈で利用される例が存在しないので詳細は省略する．

1 回事前処理. まず，拡張 Rounding LLL と同様，1 回の事前処理の後に LLL 簡約を行なう方式を考える．その後複数回の事前処理の下多項式倍高速化する方式を説明するが，拡張 Rounding LLL と違いこの改良を可能にするポイントは，事前処理の簡約を行なう行列の逆行列の計算が不要である点である．これにより，入力行

列が下三角であるという制約が不要になり，複数回の事前処理を可能にする．

任意の m の基底 $\boldsymbol{B} \in \mathbb{Z}^{n\times m}$ とパラメータ μ を入力とし，提案手法は次の 4 ステップからなる．(1) 行列 $\boldsymbol{T} = \left(\left\lceil \vec{b}_1/2^{\alpha_1-\mu} \right\rfloor | I_n\right) \in \mathbb{Z}^{n\times(n+1)}$ を生成する．(2) L^2 アルゴリズムを行列 $\boldsymbol{T}$ に適用し，その LLL 簡約行列 $\boldsymbol{S}$ を計算する．(3) $\boldsymbol{B}' = \boldsymbol{S}_{[2,n+1]} \cdot \boldsymbol{B}$ を計算する．(4) 行列 $\boldsymbol{B}'$ を入力とし L^2 アルゴリズムを適用し，その結果 $\boldsymbol{B}''$ を出力する．

行列 $\boldsymbol{T}$ は入力基底 $\boldsymbol{B}$ の 1 列目の上位 μ ビットのみを用いて構成されており，$\boldsymbol{B}_{[2,m]}$ の情報は一切用いていない．行列 T の右 $n\times n$ 部分行列が単位行列であることから，$\boldsymbol{S} = \boldsymbol{S}_{[2,n+1]} \cdot \boldsymbol{T}$ という関係は明らかに成りたち，行列 $\boldsymbol{S}_{[2,n+1]}$ は（拡張）Ronding LLL のステップ (3) の行列 $\boldsymbol{S}\cdot\boldsymbol{T}^{-1}$ と同じ役割を果たし，かつ，行列 $\boldsymbol{T}$ は逆行列を計算する必要がなくなるため下三角であるという制約が不要になる．行列 $\boldsymbol{T}$ で張られる格子の体積が $\leq \sqrt{n}\cdot 2^{\mu}$ となることから，$\|\boldsymbol{S}\| \leq c^n \cdot \sqrt{n}^{1/n} \cdot 2^{\mu/n}$ となる．行列 $\boldsymbol{T}'' = \left(\vec{t}_1 | \boldsymbol{O}_{n,m-1}\right) \in \mathbb{Z}^{n\times m}, \bar{\boldsymbol{B}} = \boldsymbol{B} - 2^{\alpha_1-\mu}\cdot\boldsymbol{T}'' \in \mathbb{Z}^{n\times m}$ と行列 $S' = (\vec{s}_1 | \boldsymbol{O}_{n,m-1})$ を定義する．行列 $\boldsymbol{B}'$ の定義から，

$$\begin{aligned}\boldsymbol{B}' &= \boldsymbol{S}_{[2,n+1]}\cdot\boldsymbol{B} \\ &= \boldsymbol{S}_{[2,n+1]}\cdot\left(2^{\alpha_1-\mu}\cdot\boldsymbol{T}'' + \bar{\boldsymbol{B}}\right) \\ &= 2^{\alpha_1-\mu}\cdot\boldsymbol{S}' + \boldsymbol{S}_{[2,n+1]}\cdot\bar{\boldsymbol{B}}\end{aligned}$$

となる．定義より，全ての $i\in[n]$ に対して $|\bar{b}_{i,1}| \leq 2^{\alpha_1-\mu}$ なので $\|\vec{\bar{b}}_1\| \leq \sqrt{n}\cdot 2^{\alpha_1-\mu}$ が成りたち，全ての $i\in[n]$ と $j\in[2,m]$ に対して $|\bar{b}_{i,j}| \leq 2^{\alpha_j}$ なので全ての $j\in[2,m]$ に対して $\|\vec{\bar{b}}_j\| \leq \sqrt{n}\cdot 2^{\alpha_j}$ が成りたつので，

$$|b'_{i,j}| \leq 2^{\alpha_1-\mu}\cdot|s'_{i,j}| + \|\boldsymbol{S}\|\cdot\|\vec{\bar{b}}_j\|$$

となり，全ての $i\in[n]$ に対して，

$$|b'_{i,1}| \leq c^n\cdot\sqrt{n}^{1/n}\cdot(1+\sqrt{n})\cdot 2^{\alpha_1-(1-1/n)\mu},$$

全ての $i\in[n]$ と $j\in[2,m]$ に対して，

$$|b'_{i,j}| \leq c^n\cdot\sqrt{n}^{1+1/n}\cdot 2^{\alpha_j+\mu/n}$$

が成りたつ．$\mu \leq \alpha_1 - \max_{j\in[2,m]}\alpha_j$ なら，

$$\max_{i\in[n],j\in[m]}|b'_{i,j}| \leq c^n\cdot\sqrt{n}^{1/n}\cdot(1+\sqrt{n})\cdot 2^{\alpha_1-(1-1/n)\mu}$$

となるので，L^2 アルゴリズムの入力長はステップ (2) で μ ビット，ステップ (4) で $\alpha_1 - (1-1/n)\mu + O(n)$ ビットになる．$\mu = \alpha_1/2$ とすると，拡張 Rounding LLL とほぼ同じ性能になる．

複数回事前処理． 前述の事前処理を 1 回行う提案手法は拡張 Rounding LLL とほぼ同じ高速化が見込めるが，入力基底に下三角であるという制約がないため，提案手法は複数回の事前処理が可能である．入力基底を $\boldsymbol{B}^{(0)} \in \mathbb{Z}^{n\times m}$ と書くこととし，k 回の事前処理を行った基底を $\boldsymbol{B}^{(k)}$ と書き，その第 j 列目の絶対値の最大値を $\alpha_j^{(k)}$ とする．行列 $\boldsymbol{B}^{(k-1)}$ を入力としたとき，事前処理の流れは 1 回事前処理と同様に次の 3 ステップからなる．(1) 行列 $\boldsymbol{T}^{(\boldsymbol{k})} = \left(\left\lceil \vec{b}_1^{(k-1)}/2^{\alpha_1^{(k-1)}-\mu(\kappa)} \right\rfloor | I_n\right) \in \mathbb{Z}^{n\times(n+1)}$ を生成する．(2) L^2 アルゴリズムを行列 $\boldsymbol{T}^{(k)}$ に適用し，その LLL 簡約行列 $\boldsymbol{S}^{(k)}$ を計算する．(3) $\boldsymbol{B}^{(k)} = \boldsymbol{S}^{(k)}_{[2,n+1]}\cdot\boldsymbol{B}^{(k-1)}$ を計算する．この事前処理を κ 回行い，最後に次の処理を行う．(4) 行列 $\boldsymbol{B}^{(\kappa)}$ を入力とし L^2 アルゴリズムを適用し，その結果 $\boldsymbol{B}'$ を出力する．この事前処理に対して，行列 $\boldsymbol{B}^{(k)}$ の成分は以下のように変化する．

補題 2 $\alpha_j^{(k)}$ を前術の定義のものとすると，全ての $j\in[2,m]$ と $k\in[0,\kappa]$ に対して，

$$\begin{aligned}\alpha_1^{(k)} &= \alpha_1 - k\left(\left(1-\frac{1}{n}\right)\mu - \log C\right) \\ \alpha_j^{(k)} &= \alpha_j + k\left(\frac{\mu}{n} + \log C\right)\end{aligned}$$

が成りたつ．ただし，$C = c^n\cdot\sqrt{n}^{1/n}\cdot(\sqrt{n}+1)$ である．

この結果は，1 回事前処理のときと同様の議論を繰り返し行うことで成立する．ここでは詳細は省略する．

アルゴリズムのパラメータを

$$\mu = \frac{\alpha_1 - \max_{j\in[2,m]}\alpha_j}{\kappa}$$

とすると，κ 回の事前処理の後全ての $j\in[m]$ に対して $\alpha_j^{(\kappa)}$ は

$$\leq \frac{\alpha_1 + (n-1)\max_{j\in[2,m]}\alpha_j}{n} + \kappa\log C$$

を満たす．

最後に，パラメータ μ と反復回数 κ の値を決め，定理 3 を証明する．各事前処理は入力長 μ の基底を入力とし，κ 回 L^2 アルゴリズムを実行する．$\mu(\kappa) = O(\alpha_1/\kappa)$ とすると，その計算量は $O(n^5\mu^2)\cdot\kappa = O(n^5\alpha_1^2/\kappa)$ となる．最後のステップ (4) の L^2 アルゴリズムの入力長は $\alpha_1^{(\kappa)} = O(\alpha_1/n+\kappa n)$ である．$\kappa = \Theta((\alpha_1/n)^{2/3})$ とすると，$\alpha_1 < n^4$ ならば $\alpha_1^{(\kappa)} = O(\kappa n) = O(\alpha_1^{2/3}n^{1/3})$ となり，事前処理とステップ (4) の L^2 アルゴリズムの計算量はいずれも $O(n^{17/3}\alpha_1^{4/3})$ となる．よって，提案手法は κ 回の事前処理を行うことで κ 倍の高速化を可能にし，反復回数 κ は入力長 $\alpha 1$ と次元 n の多項式になるため，多項式倍の高速化を可能にする．

5 実装

我々は，sage Version 6.3 を Mac OS X Version 10.9.3, 3.5 GHz Intel Core i7, 32 GB 1600 MHz DDR3 RAM で実装し，正当性を確かめる．まず，仮定 1 は我々の 30 回の実装のもと，いずれも定数 c の値は 1.02 を上回ることはなく，この値を扱う．

Chen と Nguyen の実験 [CN12] では，入力長が α で次元が n の近似 GCD 問題を解く格子 [CH12] に対する L^2 アルゴリズムの実行時間は $2.4237\log\alpha+4.5589\log n-4.5833$ ビット安全性を持つ．我々の実験では，事前処理を行うための行列 $\boldsymbol{T}$ を入力としたときの L^2 アルゴリズムの実行時間は $1.3769\log\alpha+2.5924\log n+3.9012$ ビット安全性を持つ．一般に LLL アルゴリズムの実行時間は，入力基底の Grahm-Schmidt ベクトルのノルムなどに依存して変わることがわかっており，我々が事前処理に用いる行列 $\boldsymbol{T}$ は入力長が削減できるというメリットにとどまらず，高速に L^2 アルゴリズムを実行できることがわかる．これらの評価かから見積もると，Coron らの FHE 方式 [CLT14] のパラメータは $\gamma = 15800000, \eta = 1972, \rho = 72$ で 72 ビット安全性を持つとしている．このとき，素朴に L^2 アルゴリズムを適用すると 10585 次元の格子を用いて 77.8 ビット安全性を持つが，提案方式で $\kappa = 7$ を使うことで 71.0 ビット安全性しかないと推測される．

提案手法はさらに多くの実装を重ねてより正確にその挙動を確かめることが必要ではあるが，FHE や MMP などを実現する近似 GCD 問題の安全性をより正確に見積もるために重要な技術になりうると予想される．

参考文献

[BCF+14] J. Bi, J. -S. Coron, J. -C. Faugere, P. Q. Nguyen, G. Renault, and R. Zeitoun, “Rounding and chaining LLL: Finding faster small roots of univariate polynomial congruences,” Proc. PKC 2014, LNCS 8383, pp. 185–202, Springer, 2014.

[CN12] Y. Chen, P. Q. Nguyen, “Faster Algorithms for Approximate Common Divisors: Breaking Fully-Homomorphic-Encryption Challenges over the Integers,” Proc. of Eurocrypt 2012, LNCS 7237, pp. 502–519, Springer, 2012.

[CCK+13] J. H. Cheon, J. -S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun, “Batch fully homomorphic encryption over the integers,” Proc. Eurocrypt 2013, LNCS 7881, pp. 315–335, 2013.

[CHL+14] J. H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehlé, “Cryptanalysis of the multilinear map over the integers,” Proc. Eurocrypt 2015, LNCS 9056, pp. 3–12, Springer, 2015.

[CS15] J. H. Cheon and D. Stehlé, “Fully homomorphic encryption over the integers revisited,” Proc. Eurocrypt 2015, Part I, LNCS 9056, pp. 513–536, Springer, 2015.

[CH12] H. Cohn, N. Heninger, “Approximate common divisors via lattices,” In ANTS X, 2012. IACR Cryptology ePrint Archive Report 2011/437.

[Cop96] D. Coppersmith, "Finding a Small Root of a univariate modular Equation," Proc. Eurocrypt 1996, LNCS 1070, pp. 155–165, Springer, 1996.

[CLT13] J. -S. Coron, T. Lepoint and M. Tibouchi, "Practical multilinear maps over the integers," Proc. Crypto 2013, LNCS 8042, pp. 476–493, Springer, 2013.

[CLT14] J. -S. Coron, T. Lepoint, and M. Tibouchi, "Scale-invariant fully homomorphic encryption over the integers," Proc. PKC 2014, LNCS 8383, pp. 311–328, Springer, 2014.

[CLT15] J. -S. Coron, T. Lepoint and M. Tibouchi, "New multilinear maps over the integers," IACR Cryptology ePrint Archive: Report 2015/162, 2015. To appear at Crypto 2015.

[CMNT11] J.Coron, A. Mandal, D. Naccache, M. Tibouchi, "Fully Homomorphic Encryption Over the Integers with Shorter Public Keys," Proc. Crypto 2011, LNCS 6841, pp. 487–504, Springer, 2011.

[CNT12] J.Coron, D. Naccache, M. Tibouchi, "Public Key Compression and Modulus Switching for Fully Homomorphic Encryption Over the Integers," Eurocrypt 2012, LNCS 7237, pp. 446–464, Springer, 2012.

[DGHV10] M. van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan, "Fully homomorphic encryption over the integers," Proc. Eurocrypt 2010, LNCS 6110, pp. 24–43, Springer, 2010.

[GGH13] S. Garg, C. Gentry and S. Halevi, "Candidate multilinear maps from ideal lattices," Proc. Eurocrypt 2013, LNCS 7881, pp.1–17, Springer, Heidelberg, 2013.

[Gen09] C. Gentry, "A fully homomorphic encryption using ideal lattices," In STOC '09, pp 169–178. ACM, 2009.

[How97] N. Howgrave-Graham, "Finding small roots of univariate modular equations revisited," Proc. of Cryptography and Coding, LNCS 1355, pp. 1331–142, 1997.

[How01] N. Howgrave-Graham, "Approximate integer common divisors," Proceedings of CALC 2001, LNCS 2146, pp. 51–66, Springer, 2001.

[KN15] K. Kurosawa and K. Nuida, "(Batch) fully homomorphic encryption over the integers for non-binary message spaces," Proc. Eurocrypt 2015,

[LS14] H. T. Lee, and J. H. Seo, "Security analysis of multilinear maps over the integers," Proc. Crypto 2014, LNCS 8616, pp. 224–240, Springer, 2014.

[LLL82] A. K. Lenstra, H. W. Lenstra, L. Lovasz. "Factoring polynomials with rational coefficients," Mathematische Annalen 261, pp.515–534, 1982.

[May03] A. May, "New RSA Vulnerabilities Using Lattice Reduction Methods," PhD thesis, University of Paderborn, 2003.

[NS09] P. Q. Nguyen and D. Stehlé, "An LLL algorithm with quadratic complexity," SIAM J. of Computing, 39(3):874–903, 2009.

[NV10] P. Q. Nguyen and B. Vallée, editors. "The LLL algorithm: survey and applications," Information Security and Cryptography. Springer, 2010.

[NSV11] A. Novocin, D. Stehlé, and G. Villard, "An LLL-reduction algorithm with quasi-linear time complexity: extended abstract," Proc. STOC 2011, pp. 403–412, ACM, 2011.