Gentry 準同型暗号に対する LLL 攻撃実験について

矢嶋 純 † 安田 雅哉 † 下山 武司 † 小暮 淳 †

† 株式会社富士通研究所
211-8588 川崎市中原区上小田中 4-1-1
{jyajima, myasuda, shimo}@labs.fujitsu.com
kogure@jp.fujitsu.com

あらまし 2009 年、Gentry はイデアル格子を利用した完全準同型暗号の具体的な構成法を示した。Gentry による完全準同型暗号は、暗号文に対する加算・乗算操作の演算回数が限定された準同型暗号方式 (=somewhat homomorphic encryption scheme, 本論文では限定準同型 Gentry 方式と呼ぶ) から構成される。今回、限定準同型 Gentry 方式の暗号操作可能回数と安全性の関係を検証するために、限定準同型 Gentry 方式の安全性を支える格子問題に対し、格子縮約アルゴリズムを利用した攻撃実験を行った。本論文では、代表的な格子縮約アルゴリズムの 1 つである LLL アルゴリズムを利用した攻撃実験の結果を報告する。

On the attack against Gentry's somewhat homomorphic encryption using LLL algorithm

Jun Yajima† Masaya Yasuda† Takeshi Shimoyama† Jun Kogure†

†FUJITSU LABORATORIES LTD.
1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki 211-8588, Japan
{jyajima, myasuda, shimo}@labs.fujitsu.com
kogure@jp.fujitsu.com

Abstract In 2009, Gentry proposed a concrete method for constructing a fully homomorphic encryption scheme using ideal lattices. Gentry's construction starts from a somewhat homomorphic encryption scheme, which supports limited evaluations over encrypted data. To analyze the relation between its evaluations and security, we attacked the lattice problem ensuring the security of Gentry's somewhat homomorphic encryption scheme. In this paper, we report our experimental results of attacking the lattice problem using the LLL algorithm, which is one of the typical lattice reduction algorithms.

1 はじめに

2009 年、Gentry はイデアル格子を利用した完全準同型暗号の具体的な構成法を示した [3, 4]。完全準同型暗号は、暗号化されたデータを復号せずに機密性を保ったままあらゆる処理が可能な公開鍵暗号で、クラウド・コンピューティング分野への応用が期待されている技術である [1]。Gentry による完全準同型暗号の構成では、暗号文に対する加算・乗算操作の演算回数が限定された準同型暗号方式 (somewhat homomorphic encryption scheme, 本論文では限定準同型 Gen-

try 方式と呼ぶ[1]) から構成される。ここでは、限定準同型 Gentry 方式の暗号操作可能回数と安全性の関係を考える。限定準同型 Gentry 方式は有限回の加算と乗算の両方の暗号操作が可能な方式であるが、限定準同型 Gentry 方式を支える格子問題は格子縮約アルゴリズムを利用することで解読できる可能性があり、その解読成功確率は暗号操作可能回数が多ければ多いほど増大する (cf. [5])。今回、限定準同型 Gentry 方式の安全性を保ったまま行える暗号操作可能回数を検証するために、限定準同型 Gentry 方式の安全性を支える格子問題に対し格子縮約アルゴリズムを利用した攻撃実験を行った。本論文では、代表的な格子縮約アルゴリズムの 1 つである LLL アルゴリズムを利用した攻撃実験の結果を報告する。

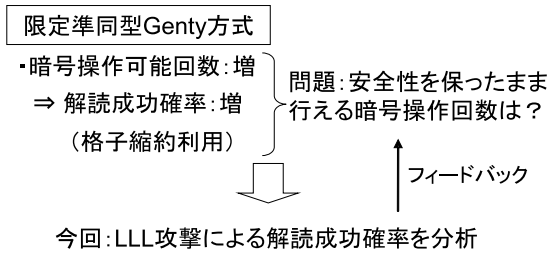

図 1: 本研究の全体像

2 限定準同型 Gentry 方式と格子縮約攻撃

ここでは、Gentry-Halevi [5] による限定準同型 Gentry 方式の構成と格子縮約アルゴリズムを用いた攻撃法を説明する。

2.1 限定準同型 Gentry 方式の構成

2 のべき数 n に対して、環 $R = \mathbb{Z}[x]/(f_n(x))$, $f_n(x) = x^n + 1$ を考える。対応

$$a_0 + a_1 x + \cdots a_{n-1} x^{n-1} \mapsto (a_0, a_1, \cdots, a_{n-1})$$

により、R と $\mathbb{Z}^n$ を同一視しておく。

- 鍵生成：パラメータ t に対し、成分 v_i が t ビット以下になる乱数ベクトル $\vec{v} = (v_0, v_1, \ldots)$ を固定する。ここで、以下の rotation 基底行列 $V = \mathrm{rot}(\vec{v})$ を考える：

$$V = \begin{bmatrix} v_0 & v_1 & v_2 & \cdots & v_{n-1} \\ -v_{n-1} & v_0 & v_1 & \cdots & v_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -v_1 & -v_2 & -v_3 & \cdots & v_0 \end{bmatrix}$$

  次に、V のエルミート標準形 $B = \mathrm{HNF}(V)$ を考える。ただし、B は以下の形であると仮定する (d は V の判別式)：

$$\begin{bmatrix} d & 0 & \cdots & 0 \\ -r & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ * & 0 & \cdots & 1 \end{bmatrix}$$

  ここで、秘密鍵を V、公開鍵を B と定める。

- 暗号化: 平文 $b \in \{0, 1\}$ に対して、ノイズベクトル $\vec{u} = (u_0, u_1, \ldots, u_{n-1})$, $u_i \in \{0, 1\}$ を選び、$\vec{a} = 2\vec{u} + b \cdot \vec{e}_1$ とおく (ただし、$\vec{e}_1 = (1, 0, \ldots, 0)$ とする。また、ベクトル $\vec{a}$ を**フレッシュ暗号文**と呼ぶことにする)。ここで、暗号文 $\vec{c}$ を

$$\vec{c} = \vec{a} \bmod B = \vec{a} - \left(\lceil \vec{a} \times B^{-1} \rfloor \times B \right)$$

  とおく (ただし、$\lceil q \rfloor$ は q の最近似整数への丸め込みとする)。行列 B の形から、$\vec{c} = (c, 0, \ldots, 0)$ となる [5, Section 5]。

- 復号: 暗号文 $\vec{c} = c \in \mathbb{Z}$ に対し、まず $\vec{a} = \vec{c} \bmod V$ を計算し、$b = \vec{a} \bmod 2$ で復号することができる (ただし、基底行列 V で生成される格子の基本領域の中にベクトル $\vec{a}$ が含まれているときに限り、復号が成功する)。

- 暗号操作: 2 つの暗号文 $c_1, c_2 \in \mathbb{Z}$ に対して、自然に加算+と乗算 $\times$ が定義できる (=$R \bmod V$ 上の演算)。

注意 (スキームの準同型性). 2 つのフレッシュ暗号文 $\vec{a}_1 = 2\vec{u}_1 + b_1 \cdot \vec{e}_1$, $\vec{a}_2 = 2\vec{u}_2 + b_2 \cdot \vec{e}_1 \in R$

[1] より正確には、加算と乗算と 2 つの演算に対して、限定された回数だけ準同型性を持つことから、“回数限定複演算準同型暗号の Gentry 方式” と呼ぶべきだが、簡略化のため、本論文ではこのように記述する。

に対して、

$$\begin{aligned}\vec{a}_1+\vec{a}_2 &= (b_1+b_2)\cdot\vec{e}_1+2(\vec{u}_1+\vec{u}_2)\in R\\ \vec{a}_1\times\vec{a}_2 &= (b_1\cdot b_2)\cdot\vec{e}_1+2(b_1\cdot\vec{u}_2+b_2\cdot\vec{u}_1)\\ &\quad +4\vec{u}_1\times\vec{u}_2\in R\end{aligned}$$

が成立するので、有限回の足し算と乗算に対して準同型性を持つことが分かる (足し算や乗算などの暗号操作を繰り返すと、フレッシュ暗号文のノイズが増大し、復号エラーが生じるため、有限回の暗号操作しかできないことに注意しておく)。

Gentry-Halevi による見積りでは、暗号操作関数 $f(x_1,\ldots,x_m)$ に対して、

$$2^t \geq C^{\deg f}\times\sqrt{M} \tag{1}$$

の条件を満たすならば、暗号操作関数 $f(x)$ に対して準同型性を持つと仮定している [5, Section 7] (ただし、M は次数 $\deg f$ の単項式の個数とする。また、C はフレッシュ暗号文のノルムに関係する値[2])。

2.2 格子縮約を利用した攻撃法

ここでは、具体的な数値例を用いて、格子縮約アルゴリズムを用いた限定準同型 Gentry 方式に対する攻撃法を説明する (秘密鍵 V を用いずに暗号文から平文を求める方法, 図 2 参照)：

例 (次元 $n=4$). $t=7$ ビット以下の係数を持つ元 $v(x)=112+98x+125x^2+77x^3\in R=\mathbb{Z}[x]/f_4(x),\ f_4(x)=x^4+1$ に対して、

$$V=\mathrm{rot}(\vec{v})=\begin{bmatrix}112 & 98 & 125 & 77\\ -77 & 112 & 98 & 125\\ -125 & -77 & 112 & 98\\ -98 & -125 & -77 & 112\end{bmatrix}$$

を考える。次に、行列 V のエルミート標準形

$$B=\mathrm{HNF}(V)=\begin{bmatrix}735969746 & 0 & 0 & 0\\ 354343231 & 1 & 0 & 0\\ 337623285 & 0 & 1 & 0\\ 147893797 & 0 & 0 & 1\end{bmatrix}$$

を計算する。平文 $b=1$ に対して、ノイズベクトル $\vec{u}=(1,0,1,1)$ を持つフレッシュ暗号文 $\vec{a}=2\vec{u}+b\cdot\vec{e}_1=(3,0,2,2)$ を考える。このとき、公開鍵 B を用いて暗号文 $\vec{c}$ を計算すると、以下のようになる：

$$\begin{aligned}\vec{c} &= \vec{a}-\left(\lceil\vec{a}\times B^{-1}\rfloor\times B\right)\\ &= (-235064415,0,0,0).\end{aligned}$$

次に、秘密鍵 V を知らずに、暗号文 $\vec{c}$ に対応する平文 b を求める方法を以下に示す：公開鍵 B と暗号文 $\vec{c}$ のみで構成される行列

$$C=\begin{bmatrix}B & {}^t\vec{0}\\ \vec{c} & 1\end{bmatrix}=\begin{bmatrix}735969746 & 0 & 0 & 0 & 0\\ 354343231 & 1 & 0 & 0 & 0\\ 337623285 & 0 & 1 & 0 & 0\\ 147893797 & 0 & 0 & 1 & 0\\ -235064415 & 0 & 0 & 0 & 1\end{bmatrix}$$

を考える (行列の最右下の成分はある小さい値で十分)。このとき、格子基底行列 C に対して、格子縮約アルゴリズムである LLL 基底アルゴリズムを適用すると、

$$\begin{bmatrix}3 & 0 & 2 & 2 & 1\\ -29 & 41 & 41 & -25 & 58\\ -11 & 98 & 43 & -5 & -41\\ -32 & -91 & 75 & -14 & -32\\ 102 & -14 & -31 & -106 & -29\end{bmatrix}$$

となる。ここで、第 1 行目のベクトル $(3,0,2,2,1)$ から、$\vec{c}$ に対応するフレッシュ暗号文 $\vec{a}$ を求めることができるので、平文 $b=\vec{a}\bmod 2=1$ が漏洩する。

2.3 暗号操作可能回数 vs 格子縮約攻撃

限定準同型 Gentry 方式は有限回の加算と乗算の両方の暗号操作が可能な方式であるが、その暗号操作可能回数は関係式 (1) から見積もることができる。ここでは、乗算回数と鍵成分パラメータ t の関係を表 1 にまとめておく[3]:

一方、鍵成分パラメータ t の値が次元 n に比べて十分大きいとき、限定準同型 Gentry 方式

[2] フレッシュ暗号文のノイズベクトル $\vec{u}$ の取り方に大きく依存するが、Gentry-Halevi は実験的に $C\approx 7$ と求めている ($||\vec{u}||=\sqrt{15}\sim\sqrt{20}$ のとき) [5, Section 10]。

[3] Gentry-Halevi と同じように $C\approx 7$ と計算した。つまり、ここでは乗算回数 m に対して $2^t\geq 7^{m+1}$ が成立すると仮定した。

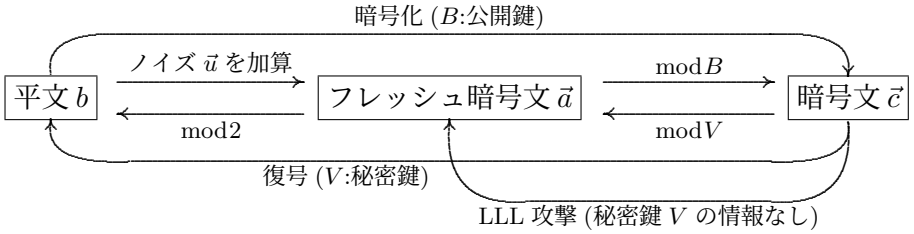

図 2: 限定準同型 Gentry 方式と LLL 攻撃

表 1: 乗算回数と鍵成分パラメータ t の関係

乗算回数	t
1	$6, 7, 8$
2	$9, 10, 11$
3	$12, 13, 14$
4	$15, 16$
5	$17, 18, 19$

の安全性を支える格子問題は §2.2 で説明した格子縮約攻撃によって解読される可能性がある[4]。本論文では、LLL アルゴリズムを利用した攻撃を用いることで限定準同型 Gentry 方式の安全性が崩れる (n, t) の値を実験的に求めた。

- 暗号操作可能回数
  - 鍵成分パラメータ t に関係 (表 1 参照)
- 限定準同型 Gentry 方式の安全性
  - 鍵成分パラメータ t と次元 n に関係 (LLL 攻撃に対する安全性は図 3 参照)

3 攻撃実験

ここでは、今回行った限定準同型 Gentry 方式に対する攻撃実験に関して説明する。

3.1 準備

今回の実験を行うにあたり、OS は CentOS とし、行列演算やベクトル演算は NTL ライブラリ (`http://www.shoup.net/ntl/`) を用いて行った。また、多倍長の演算については GMP ライブラリ (`http://gmplib.org/`) を用いた。開発言語は C++とした。コンパイル環境、ライブラリについて以下に示す。

- コンパイル環境
  - コンパイラ: gcc version 4.1.2 20080704 (Red Hat 4.1.2-48)
  - ライブラリ: NTL 5.5.2(2009.08.14) GMP ライブラリ 5.0.1

また、実験を行う計算機環境としては、下記の計算機 A, B, C を各 10 台ずつ準備した。

- 実験環境

  計算機 A
  - CPU: Intel Xeon X3460 (2.80GHz)
  - OS: CentOS 5.5 x86_64
  - Memory: 8GB

  計算機 B
  - CPU: Intel Core2 Quad Q9650 (3.00GHz)
  - OS: CentOS 5.4 x86_64
  - Memory: 4GB

  計算機 C
  - CPU: Intel Core2 Quad Q9450 (2.66GHz)
  - OS: CentOS 5.5 x86_64
  - Memory: 4GB

3.2 実装アルゴリズム

本実験を行うにあたって実装したアルゴリズムの概略を Algorithm 1 に示す。

[4]理論的な背景はここでは省略することにする。

NTL ライブラリにてデータをランダムに生成するには、まず SetSeed にてシード値をセットした後、RandomBit_ZZ などの関数を呼び出す。SetSeed することにより、生成される擬似乱数系列が決定される。本実験では、鍵シードと平文シードを別々に準備してそれぞれセットすることで、同一鍵による複数計算機での実験を可能にしている。

Algorithm 1 攻撃実験アルゴリズム．{} 内は NTL ライブラリの関数名

Input: 次元 n, 鍵成分パラメータ t, 実験回数 ℓ, ノイズベクトルのハミングウェイト h, 鍵シード $s1$, 平文/暗号文シード $s2$

Output: LLL 攻撃成功回数 $success$

1: 擬似乱数生成用に $s1$ をシードとしてセットし、擬似乱数系列を決定する {SetSeed($s1$)};
2: (n, t) から秘密鍵行列 V と公開鍵行列 B を 1 でセットした鍵シードに基づいてランダムに生成 (§2.1 参照);
3: 擬似乱数生成用に $s2$ をシードとしてセットし、擬似乱数系列を決定する {SetSeed($s2$)};
4: **for** $i = 1$ to ℓ **do**
5: $plain$(0 or 1) を 3 でセットした平文/暗号文シードに基づいてランダムに生成 {RandomBits_ZZ(1)};
6: ハミングウェイト h を持つノイズベクトル $\vec{u}$ を 3 でセットした平文/暗号文シードに基づいてランダムに生成し、フレッシュ暗号文 $\vec{a} = 2\vec{u} + plain \cdot \vec{e}_1$ を生成。暗号文 $\vec{c} = \vec{a} \bmod B$ を生成;
7: B, $\vec{c}$ を元に行列 C を作り、C に対して LLL を行った結果の第一行目が $\vec{a}$ と同一であることを比較 (§2.2 参照)。同一であれば $ret = true$、そうでなければ $ret = false$ とする;
8: **if** $ret == true$ **then**
9: $success = success + 1$;
10: **end if**
11: **end for**
12: Output $success$

3.3 実験内容

本実験で扱うパラメータ一覧を表 2 に示す。平文、ノイズベクトルは各実験毎に毎回生成した。鍵については各次元毎に 1 パターンずつ準備した。また、LLL を行う際のリダクションパラメータ $\delta = 0.99$ とした。

表 2: 実験パラメータ

次元 n	鍵成分パラメータ t	実験回数 ℓ
128	4, 5, 6, 7, 8, 9	各 1000 回
256	4, 5, 6, 7, 8, 9, 10	各 1000 回
512	6, 7, 8, 9, 10, 11, 12	各 100 回

注意 (使用した LLL 関数について). NTL ライブラリには整数版 LLL の “LLL” の他に、浮動小数点版の “G_LLL_XD” や、“G_LLL_RR” などが存在する。予備実験を行った結果、今回の実験パラメータの範囲においては G_LLL_XD が高速であることが判明し、結果も LLL と同一となったため、G_LLL_XD を採用することとした。ただし、$n = 512$ の $t \geq 9$ の範囲においては精度の問題で実験を完了できなかったため (Warning が出て終了してしまう)、この範囲においては整数版の LLL を使用した。

3.4 実験結果

- 攻撃実験成功確率：

  §3.3 で示したパラメータについて実験を行い、それぞれ成功確率を導出した。結果を図 3 に示す。

- 処理時間：

  表 2 の範囲のパラメータについて、計算機 A において LLL1 回に要する平均時間 (Algorithm 1 の 7 に要する時間) を測定した。それぞれ 128 次元、256 次元についてはそれぞれ 10 回実行した平均、512 次元については時間の都合上、1 回実行した結果である。512 次元の鍵成分パラメータ 10 が、他と比較して相当低速になっているが、これは使用している関数が G_LLL_XD ではなく、整数版 LLL だからであると考えられる。

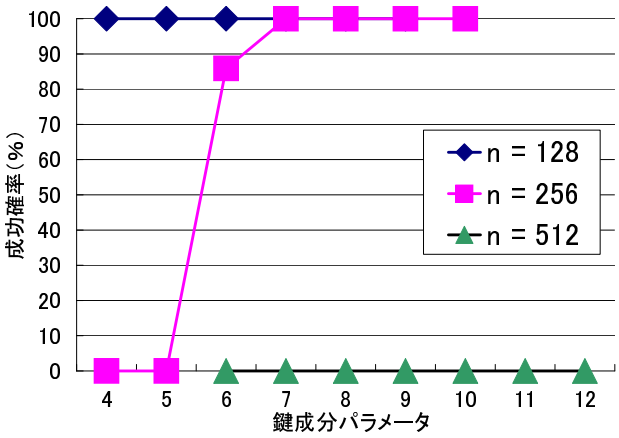

図 3: 攻撃実験成功確率

表 3: 処理時間 (秒)

	$t = 6$	$t = 8$	$t = 10$
$n = 128$	129.8	163.4	223.2
$n = 256$	1764.6	2122.8	2815.0
$n = 512$	15336.0	28284.2	87691.3

3.5 考察

今回の攻撃実験から (図 3 参照)、

- $n = 128$ のとき、$t \geq 4$ で 100%攻撃成功
- $n = 256$ のとき、$t \geq 7$ で 100%攻撃成功

であることが分かった。この結果から、$n \leq 256$ における限定準同型 Gentry 方式で行える暗号操作回数は、表 1 から乗算 1 回以下であるため、複雑な処理には適していないことが分かった。$n = 512$ に対しては、今回の攻撃実験では、攻撃成功確率が 100%になる鍵成分パラメータ t を求めることはできなかったが、$t = 15 \sim 17$ 程度で 100%になると予想している (今後の課題)。

4 まとめ

今回、限定準同型 Gentry 方式の安全性を保ったまま行える暗号操作可能回数を検証するために、$n = 128, 256, 512$ 次元の限定準同型 Gentry 方式に対して LLL 攻撃実験を行い、その実験結果を報告した。今回の実験結果から、$n \leq 256$ における限定準同型 Gentry 方式では、乗算 1 回程度しか暗号操作を行えないことを示すことができた[5]。今回の攻撃実験では、$n = 512$ 以上の次元に対する実験が不十分だった。よって、今後の課題としては、より高次元における攻撃実験を行うことで、$n = 512, 1024$ 程度の限定準同型 Gentry 方式の暗号操作可能回数を見極めることである。さらには、LLL よりも精度の高い格子縮約アルゴリズムを用いた攻撃実験も今後行う予定である。

参考文献

[1] IBM プレスリリース, `http://www-03.ibm.com/press/us/en/pressrelease/27840.wss`.

[2] N. Gama and P. Q. Nguyen, “Predicting lattice reduction”, In *Advances in Cryptology - EUROCRYPT 2008*, Springer LNCS 4965, 31-51, 2008.

[3] C. Gentry, “Fully homomorphic encryption using ideal lattices”, In *Symposium on Theory of Computing - STOC 2009*, ACM, 169-178, 2009.

[4] C. Gentry, “A fully homomorphic encryption scheme”, Manuscript, 2009.

[5] C. Gentry and S. Halevi, “Implementing Gentry’s fully-homomorphic encryption scheme”, In *Advances in Cryptology -EUROCRYPTO 2011*, Springer LNCS 6632, 129-148, 2011.

[6] P. Q. Nguyen and B. Vallée, The LLL Algorithm: Survey and Applications, Springer, 2010.

[5] LLL より精度の高い格子縮約アルゴリズムを用いた攻撃も考慮すれば、乗算 1 回の暗号操作もできない可能性がある