A New Algorithm for Solving the Approximate Common Divisor Problem and Cryptanalysis of the FHE based on GACD

Jintai Ding[1,3] and Chengdong Tao[2]

[1] Chongqing University and University of Cincinnati
[2] South China University of Technology
[3] Corresponding Author
jintai.ding@gmail.com, chengdongtao2010@gmail.com

Abstract. In this paper, we propose a new algorithm for solving the approximate common divisors problems, which is based on LLL reduction algorithm of certain special lattice and linear equation solving algorithm over integers. Through both theoretical argument and experimental data, we show that our new algorithm is a polynomial time algorithm under reasonable assumptions on the parameters. We use our algorithm to solve concrete problems that no other algorithm could solve before. Further more, we show that our algorithm can break the fully homomorphic encryption schemes, which are based on the approximate common divisors problem, in polynomial time in terms of the system parameter λ.

Key words. Approximate common divisors problems; Fully homomorphic encryption; Lattice

1 Introduction

Approximate common divisors problems including partial approximate common divisors(PACD) problem and general approximate common divisors(GACD) problem were first introduced by Howgrave-Graham in [12]. In this paper, we only consider the GACD problem, since PACD problem is a special case of GACD problem. Therefore, our algorithm can be used to solve PACD problem without any change. General approximate common divisors(GACD) problem is defined as follows:

For a set of parameters γ ,η, and ρ, given **polynomial (in γ η, and ρ)** *many different integers in the form: $x_i = pq_i + r_i (i = 1, \ldots, n)$, the problem is to recover p, where p and $q_i, (i = 1, \ldots, n)$ are very large integers, x_i are of bit length γ and p is of bit length η, and $r_i (i = 1, \ldots, n)$ are small integers with the bit length no more than ρ. Here r_i are called the error terms*

Since in applications ρ is much smaller than η, in this paper, we will consider mainly the case, where $\rho < \eta/2$. Therefore, we assume $\rho < \eta/2$ if it is not otherwise specified.

The hardness of solving this problem for small p (relative to the size of x_i) and small error terms (relative to the size of p) was recently proposed as the foundation for a fully homomorphic cryptosystem. At EUROCRYPT'10, Van Dijk et al. proposed a fully homomorphic encryption (FHE) scheme based on the hardness of GACD problem [16]. At CRYPTO'11, Coron et al. presented a more efficient variant of the FHE scheme in [6] which was based on PACD problem.

A simple approach for solving GACD problem is exhaustive search on the error terms. If r_i are sufficiently small, namely if $|r_i| < B$, where B is a fixed small integer, then we can find p by exhaustive search, i.e., one can trying every r_1 and r_2 and check whether $gcd(x_1 - r_1, x_2 - r_2)$ is sufficiently large and eventually recover p. The state of the art algorithm for computing GCD's is the Stehlè-Zimmermann algorithm with time complexity $O(\gamma)$ for integers of γ bits[17]. Therefore, the time complexity of solving GACD problem by exhaustive search on the error terms is $O(2^{2\rho}\gamma)$.

In EUROCRYPT'12, Chen and Nguyen gave an algorithm which provides an exponential speedup over exhaustive search to solve approximate common divisors problem [4], which is essentially based a clever exhaustive search on the error terms through certain polynomials. However, their approach requires large memory. For their algorithm, they only need 2 elements in the set of x_i and the complexity is given as $O(2^{\frac{3}{2}\rho}\gamma)$. This means, if γ is around 2^{20} and any ρ bigger than 40, their algorithm would be considered infeasible.

In [12], Howgrave-Graham also gives a lattice approach to solve two elements GACD problem. This approach is related to Coppersmith's algorithm for finding small solutions to univariate and bivariate modular equations. When $\frac{\rho}{\gamma}$ is smaller than $(\frac{\eta}{\gamma})^2$, this approach recovers p. However, when ρ, η, γ do not satisfy the constraint, the approach does not degrade gracefully. Furthermore, in [5], Cohn and Heninger analyze the multivariate generalization of Howgrave-Graham's algorithm for the GACD problem by using many x_i. In this algorithm, the GACD problem used in cryptography is reduced to running the LLL algorithm on a lattice basis of high dimension and large entries to directly find all the error terms r_i. However, in [4], they show that the Cohn-Heninger attack on the FHE challenges in [6] is actually slower than exhaustive search on the challenges, and therefore much slower than the attack in [4].

1.1 The contribution of this paper

In this paper, we propose a polynomial time algorithm for solving the GACD problem in terms of $\gamma, \eta, , \rho$ under the assumption that $\rho < \eta/2$.

The main ideal of our method for solving GACD problem is to reduce the problem first to a special lattice reduction problem, but unlike the case of [5], we will not be able to find r_i directly, but rather the results of the reduction allow us to find many linear equations satisfies by r_i. Then we recover those r_i through solving those integer equations with the help of the bound of r_i and LLL algorithm. Then we can recover p via Euclidean algorithm.

The algorithm can be summarized as follows:

1. We first randomly select an positive integer $N \in (2^{\gamma-1}, 2^{\gamma})$, and an appropriate small positive number t .
2. We construct a lattice $\mathcal{L}_1$ spanned by rows of the following matrix:

$$\begin{pmatrix} 1 & & & x_1 \\ & 1 & & x_2 \\ & & \ddots & \vdots \\ & & 1 & x_t \\ & & & -N \end{pmatrix}.$$

   Let $v \in \mathcal{L}_1$,then v has the form $v = (u_1, \ldots, u_t, \sum_{i=1}^{t} u_i \cdot x_i - N u_{t+1})$, where $u_1, \ldots, u_{t+1}$ are integers. Thus the length of vector v in Euclid norm is

$$\|v\| = \sqrt{\sum_{i=1}^{t} u_i^2 + (\sum_{i=1}^{t} u_i x_i - u_{t+1} N)^2}.$$

3. We apply LLL lattice reduction algorithm with $\delta = \frac{3}{4}$ to find a short vector and it turns out (if t is big enough) that this short vector gives solution to the equation:

$$\sum_{i=1}^{t} u_i \cdot r_i = \sum_{i=1}^{t} u_i \cdot x_i, \tag{1}$$

which implies that $N \mid p\sum_{i=1}^{t} u_i \cdot q_i$; and the short vector also satisfies the condition:

$$\frac{N}{2} > \left|\sum_{i=1}^{t} u_i \cdot r_i\right|. \tag{2}$$

The key point here is that we need to make sure that we choose t to be large enough such that the relations (2) and (1) are true. By now, we still do not know how to chose the best t, but through experiments, we can show that t is bounded by linear function of γ, η, ρ.
To us, even more amazing thing is that we still do not know why the LLL algorithm should gives us solution satisfying those nice properties, though we have intuitive explanation why this happens, which will be presented in the next section.

4. The LLL reduction in general gives us $t - z$ such vectors, where z is a small positive integer (≤ 2), each vector gives us a linear equation satisfied by $r_1, \ldots, r_t$. We find the integer solutions of this equations by solving the derived linear system in integers. The integer solutions can be expressed as follow:
$$\mathbf{d} = \mathbf{d_0} + t_1\mathbf{d_1} + ... + t_z\mathbf{d_z},$$
where $\mathbf{d_0}$ is a special solution of the linear system, $t_1, ..., t_z$ are integers, $\mathbf{d_1}, ..., \mathbf{d_z}$ is a basis of integer solution space of the corresponding homogeneous linear equations.
5. We construct a lattice spanned by the row vectors $\mathbf{d_0}, \mathbf{d_1}, ..., \mathbf{d_z}$. Obviously, $(r_1, \ldots, r_t)$ is a short vector of the lattice. Thus we can find $r_1, \ldots, r_t$ by LLL algorithm.
6. Finally, we recover the common divisor p by Euclidean algorithm, that is $p = gcd(x_1 - r_1, x_2 - r_2)$.

With assumptions supported by experimental data, we can show that our method is polynomial time in terms of γ, η, ρ, since the main time consumption step is the LLL lattice reduction algorithm with $\delta = \frac{3}{4}$.

We show that we can break the fully homomorphic encryption (FHE) scheme which were proposed in [16][6] in polynomial time, where $\tau = n$, is associate to the bit length of x_i and r_i.

When n in is very small, i.e., $n = 2$, our algorithm is invalid.

In this paper, we will first present the algorithm in details and its complexity analysis with support from experimental data. Then we will show how this algorithm can be used to attack the FHE based on GACD in polynomial time.

2 The new algorithm for GACD

We will first present some background facts we need.

2.1 Background on Lattice

We present some facts on lattice which would be used in next subsection. More details can be found in [14] and [15].

Let $\mathbb{R}^m$ be the m-dimensional Euclidean space. Let $\mathbb{Z}$ be the set of integer.

Definition 1 [14]. A lattice in $\mathbb{R}^m$ is the set

$$\mathcal{L}(\mathbf{b_1}, ..., \mathbf{b_n}) = \{\sum_{i=1}^{n} x_i\mathbf{b_i} : x_i \in \mathbb{Z}\}$$

of all integral combinations of n linearly independent row vectors $\mathbf{b_1}, ..., \mathbf{b_n}$ in $\mathbb{R}^m$ $(m \geq n)$. Equivalently, if we define B as the $m \times n$ matrix whose rows are $\mathbf{b_1}, ..., \mathbf{b_n}$, then the lattice generated by B is

$$\mathcal{L}(B) = \{xB : x \in \mathbb{Z}^n\}.$$

The integers n and m are called the rank and dimension of the lattice, respectively. The sequence of vectors $\mathbf{b_1}, ..., \mathbf{b_n}$ is called a lattice basis.

Definition 2 [14]. For any lattice basis B we define the half parallelepiped

$$\mathcal{P}(B) = \{xB | x \in \mathbb{R}^n : \forall i, 0 \leq x_i < 1\}.$$

Definition 3 [14]. The determinant of a lattice $\mathcal{L}$, denoted $\det(\mathcal{L})$, is the n-dimension volume of the fundamental parallelepiped $\mathcal{P}(B)$ spanned by the basic vectors. In symbols, this can be written as $\det(\mathcal{L}) = \sqrt{|\det(BB^T)|}$. In the special case that B is a square matrix, and we have $\det(\mathcal{L}) = |\det(B)|$.

Definition 4 [15]. Let $\mathcal{L}$ be a lattice of rank n. we define the ith successive minimum as

$$\lambda_i = inf\{r | dim(span(\mathcal{L} \cap B_m(0, r))) \geq i\}, (i = 1, ..., n),$$

where $B_m(0, r) = \{x \in \mathbb{R}^m : \|x\| \leq r\}$ is the closed ball of radius r around 0.

The volume of $B_m(0, r)$ is $vol(B_m(0, r)) = r^m \frac{\pi^{m/2}}{\Gamma(m/2+1)}$, where

$$\Gamma(x) = \int_0^\infty y^{x-1} e^{-y} dy$$

is a Gamma function. Let $\overline{B_m(0, r)}$ denote the closure of $B_m(0, r)$.

Theorem 1[15]. Let $\mathcal{L}$ be a full rank lattice with rank n, then

$$\lim_{r \to \infty} \frac{r^n vol(B_n(0, 1))}{\left|\overline{B_n(0, r)} \bigcap \mathcal{L}\right|} = det(\mathcal{L}),$$

where $\left|\overline{B_n(0, r)} \bigcap \mathcal{L}\right|$ is the number of lattice vectors contained in $\overline{B_n(0, r)}$. Rewrite this limit shows that, heuristically:

$$\frac{r^n vol(B_n(0, 1))}{det(\mathcal{L})} \approx \left|\overline{B_n(0, r)} \bigcap \mathcal{L}\right|.$$

Theorem 2 [15]. Let $\mathcal{L}$ be a lattice of rank n with successive minimal vectors $\lambda_1(\mathcal{L}), \ldots, \lambda_n(\mathcal{L})$. Let $\mathbf{a_1}, ..., \mathbf{a_n}$ be an LLL-reduced basis with factor $\delta = \frac{3}{4}$ of a lattice $\mathcal{L}$ in $\mathbb{R}^m$. Then

1. $\|\mathbf{a_1}\| \leq 2^{\frac{n-1}{4}} det(\mathcal{L})^{\frac{1}{n}}$.
2. $\|\mathbf{a_i}\| \leq 2^{\frac{n-1}{2}} \lambda_i(\mathcal{L})$, $i = 1, ..., n$.
3. $\prod_{i=1}^{n} \|\mathbf{a_i}\| \leq 2^{\frac{n(n-1)}{4}} det(\mathcal{L})$.

Theorem 3 [15]. The LLL basis reduction algorithm with factor $\delta = \frac{3}{4}$ computes an LLL-reduced basis in polynomial time in the maximal bit-length of the coefficients of the input basis, the lattice rank n, and the space dimension m. Specifically, if $\mathbf{b_1}, ..., \mathbf{b_n}$ is an input lattice basis, $C = \max\{\|\mathbf{b_1}\|, ..., \|\mathbf{b_n}\|\}$, then LLL runs in $O\big(n^5 m (\log_{\frac{4}{3}} C)^3\big)$ bit operations, under school-multiplication.

2.2 The GACD algorithm

In this section, we describe our algorithm for the GACD problem. We consider the first t integers $x_1, \ldots, x_t$. We start from the first observation that is needed for explaining our algorithm.

Lemma 1. For any $x_i = pq_i + r_i$ and any positive integer $N > 2^{\rho+1}$, $(x_i \mod N) = r_i$ if and only if $N | pq_i$.

Proof. It is evident.

Lemma 2. Let $x_i = pq_i + r_i (i = 1, \ldots, t)$. Let $u_i (i = 1, \ldots, t)$ be t integers and N be positive integer satisfy $N > 2\left|\sum_{i=1}^{t} u_i \cdot r_i\right|$. Then

$$\sum_{i=1}^{t} u_i \cdot r_i = (\sum_{i=1}^{t} u_i x_i \mod N)$$

if and only if $N \mid p\sum_{i=1}^{t} u_i \cdot q_i$.

Proof. Since $N > 2\left|\sum_{i=1}^{t} u_i \cdot r_i\right|$, on the one hand, if $N \mid p\sum_{i=1}^{t} u_i \cdot q_i$, since $\sum_{i=1}^{t} u_i \cdot x_i = p\sum_{i=1}^{t} u_i \cdot q_i + \sum_{i=1}^{t} u_i \cdot r_i$, therefore $\sum_{i=1}^{t} u_i \cdot r_i = (\sum_{i=1}^{t} u_i \cdot x_i \ mod\ N)$. On the other hand, if $\sum_{i=1}^{t} u_i \cdot r_i = (\sum_{i=1}^{t} u_i \cdot x_i \ mod\ N)$, then $N \mid (\sum_{i=1}^{t} u_i \cdot x_i - \sum_{i=1}^{t} u_i \cdot r_i)$, therefore $N \mid p\sum_{i=1}^{t} u_i \cdot q_i$.□

Theorem 4. Let $x_i = pq_i + r_i (i = 1, \ldots, t)$ be be t integers and N be a positive number. Let $S = \{(u_1, \ldots, u_t) : 2|\sum_{i=1}^{t} u_i \cdot r_i| < N, u_i \in \mathbb{Z}, i = 1, ..., t\}$ and $|S|$ be the number of elements in S. If $|S| > N$, then there exists at least a element in $u \in S$ such that

$$\sum_{i=1}^{t} u_i \cdot r_i = (\sum_{i=1}^{t} u_i x_i \mod N)$$

with highly probability.

Proof. It is easy to see that for any integer z , the probability of z divisible by N is $\frac{1}{N}$. Thus if we randomly choose more than N integers, there is an integer divisible by N with highly probability(approximate 1). Since $|S| > N$, there exists an element in $u \in S$ such that $N \mid p\sum_{i=1}^{t} u_i \cdot q_i$ with highly probability. Therefore, there exists an element in $u \in S$ such that

$$\sum_{i=1}^{t} u_i \cdot r_i = \sum_{i=1}^{t} u_i x_i \mod N$$

with highly probability.□

Theorem 4 implies that by collecting many vectors in the set S which satisfy (1) and (2), we can obtain a linear equations satisfied by $r_1, \ldots, r_t$. To find such a u, we will use LLL lattice reduction.

We first build a lattice $\mathcal{L}_1$ spanned by rows of the following matrix:

$$B_1 = \begin{pmatrix} 1 & & & x_1 \\ & 1 & & x_2 \\ & & \ddots & \vdots \\ & & 1 & x_t \\ & & & -N \end{pmatrix}. \tag{3}$$

where N is an appropriate integer. We show that, for two appropriate positive integers N and t, there exists at least one short vector $u \in \mathcal{L}_1$ satisfy (1) with highly probability through experiments.

Theorem 5. Let $\mathcal{L}_2$ be a lattice spanned rows of the following matrix:

$$B_2 = \begin{pmatrix} 1 & & & r_1 \\ & 1 & & r_2 \\ & & \ddots & \vdots \\ & & 1 & r_t \end{pmatrix}. \tag{4}$$

then $\mathcal{L}_1 \bigcap \mathcal{L}_2 \neq \emptyset$ if and only if there exist $u_1, ..., u_t \in \mathbb{Z}$ such that $N \mid p \sum_{i=1}^{t} u_i \cdot q_i$.

Proof. The vector in lattice $\mathcal{L}_1$ has the form: $(\mu_1, ..., \mu_t, \sum_{i=1}^{t} \mu_i \cdot x_i - \mu_{t+1}N)$, where $\mu_1, ..., \mu_{t+1} \in \mathbb{Z}$. The vector in lattice $\mathcal{L}_2$ has the form: $(\nu_1, ..., \nu_t, \sum_{i=1}^{t} \nu_i \cdot r_i)$, where $\nu_1, ..., \nu_t \in \mathbb{Z}$.

On one hand, suppose that $v \in \mathcal{L}_1 \bigcap \mathcal{L}_2$, then there exist $\bar{\mu}_1, ..., \bar{\mu}_{t+1} \in \mathbb{Z}$ and $\bar{\nu}_1, ..., \bar{\nu}_t \in \mathbb{Z}$ such that

$$(\bar{\mu}_1, ..., \bar{\mu}_t, \sum_{i=1}^{t} \bar{\mu}_i \cdot x_i - \bar{\mu}_{t+1}N) = (\bar{\nu}_1, ..., \bar{\nu}_t, \sum_{i=1}^{t} \bar{\nu}_i \cdot r_i).$$

Therefore $\bar{\mu}_i = \bar{\nu}_i, (i = 1, ..., t)$ and $\sum_{i=1}^{t} \bar{\mu}_i \cdot x_i - \bar{\mu}_{t+1}N = \sum_{i=1}^{t} \bar{\nu}_i \cdot r_i$. That is $\sum_{i=1}^{t} \bar{\mu}_i \cdot x_i - \bar{\mu}_{t+1}N = \sum_{i=1}^{t} \bar{\mu}_i \cdot r_i$. Denote $u_i = \bar{\mu}_i = \bar{\nu}_i, (i = 1, ..., t)$, from Lemma 2, we have $N \mid p \sum_{i=1}^{t} u_i \cdot q_i$.

On the other hand, if there exist $u_1, ..., u_t \in \mathbb{Z}$ such that $N \mid p \sum_{i=1}^{t} u_i \cdot q_i$, then there is a integer u_{t+1} such that $p \sum_{i=1}^{t} u_i \cdot q_i = Nu_{t+1}$. Thus $p \sum_{i=1}^{t} u_i \cdot q_i - Nu_{t+1} = 0$. Therefore $\sum_{i=1}^{t} u_i \cdot x_i - Nu_{t+1} = \sum_{i=1}^{t} u_i \cdot r_i$. Then $(u_1, ..., u_t, \sum_{i=1}^{t} u_i \cdot r_i) \in \mathcal{L}_1 \bigcap \mathcal{L}_2$.□

Theorem 6. If $N < |\overline{B(0,k)} \bigcap \mathcal{L}_2|$, where $k = \sqrt{t+1} det(\mathcal{L}_1)^{\frac{1}{t+1}}$, then there exists at least a $v = (u_1, ..., u_{t+1})$ such that $v \in \overline{B(0,k)} \bigcap \mathcal{L}_2 \bigcap \mathcal{L}_1$ with highly probability. Moreover, we have

$$\sum_{i=1}^{t} u_i \cdot r_i = \sum_{i=1}^{t} u_i x_i \mod N.$$

Proof. Since $N < |\overline{B(0,k)} \bigcap \mathcal{L}_2|$, from Theorem 4, there exists at least a vector $v = (u_1, ..., u_{t+1}) \in \overline{B(0,k)} \bigcap \mathcal{L}_2$ such that

$$\sum_{i=1}^{t} u_i \cdot r_i = \sum_{i=1}^{t} u_i x_i \mod N.$$

with highly probability. Thus by Lemma 2, we have $N \mid p\sum_{i=1}^{t} u_i \cdot q_i$. Therefore, from Theorem 5, we have $v \in \overline{B(0,k)} \bigcap \mathcal{L}_2 \bigcap \mathcal{L}_1$ $\square$

Theorem 6 implies that v is a short vector of lattice $\mathcal{L}_1$ satisfies

$$\|v\| \leq k = \sqrt{t+1} det(\mathcal{L}_1)^{\frac{1}{t+1}}.$$

Therefore, we can try to find v by using LLL lattice reduction algorithm.

Now, let us find a way to choose the right N and t for our algorithm.

First, we choose $N \in (2^{\gamma-1}, 2^{\gamma})$, since N is expected to close to x_i . In the following, we try to find a lower bound of t.

When we do LLL, the shortest vector will be roughly of length $2^{\frac{t}{4}+\frac{\gamma}{t+1}}$. For the short vector, the first t coordinate should be of size $\frac{1}{\sqrt{t+1}} 2^{\frac{t}{4}+\frac{\gamma}{t+1}}$. First question one would ask is that if there exists a short vector that satisfying the condition (1). The vector in lattice $\mathcal{L}_1$ has the form:

$$(\mu_1, ..., \mu_t, \sum_{i=1}^{t} \mu_i \cdot x_i - \mu_{t+1} N),$$

and this means that

$$\mu_i \leq \frac{1}{\sqrt{t+1}} 2^{\frac{t}{4}+\frac{\gamma}{t+1}},$$

for $i < t+1$. We can imagine that we will do a search of the μ_i, $i < t+1$ in the range above for a short vector, which has a total size of

$$(2\frac{1}{\sqrt{t+1}} 2^{\frac{t}{4}+\frac{\gamma}{t+1}})^t = \frac{1}{\sqrt{t+1}^t} 2^{t+\frac{t^2}{4}+\frac{t\gamma}{t+1}}.$$

From Theorem 6, we know that if this number is bigger than N, then we have a high probability to have a vector such that the last coordinate satisfying the condition (1). This mean we should have that

$$\frac{1}{\sqrt{t+1}^t} 2^{t+\frac{t^2}{4}+\frac{t\gamma}{t+1}} \geq N \approx 2^{\gamma}.$$

This means that

$$t + \frac{t^2}{4} + \frac{t\gamma}{t+1} - t\log_2(t+1) \geq \gamma,$$

therefore

$$t^2/4 \geq \frac{\gamma}{t+1},$$

which essentially means that

$$t \geq (4\gamma)^{1/3}.$$

However, the interesting part is that in experiments, we need much smaller t than this bound, which we can not explain.

What puzzles us even more is why for a short vector, when t is big enough, the equation (1) should be true. Our only explanation is that when

$$\mu_i \leq \frac{1}{\sqrt{t+1}} 2^{\frac{t}{4}+\frac{\gamma}{t+1}},$$

for $i < t+1$, the last coordinate is given by

$$\sum_{i=1}^{t} \mu_i \cdot x_i - \mu_{t+1} N =$$
$$\sum_{i=1}^{t} \mu_i \cdot (pq_i + r_i) - \mu_{t+1} N =$$
$$p \sum_{i=1}^{t} \mu_i \cdot q_i - \mu_{t+1} N + \sum_{i=1}^{t} \mu_i \cdot r_i.$$

Since ρ is small, the last part of summation of $\mu_i r_i$ are of number which is of size $2^{\rho+\frac{\gamma}{t+1}} << N$, which therefore are insignificant in the sense the first summation are really large number, which is of size $2^{\gamma+\frac{\gamma}{t+1}}$ and they dominant the computations and the LLL tries to make this part to be zero while essentially ignore the last summation since they are too small comparatively. Therefore we could achieve the relation (1). Surely this is only a heuristic explanation and a theoretical proof will be a very significant result.

The GACD algorithm is showed as follow:

The GACD algorithm .
Input: A appropriate positive integers t and $x_1, \ldots, x_t$.
Output: Integer p .

1. Randomly choose $N \in (2^{\gamma-1}, 2^{\gamma})$.
2. Reduce lattice $\mathcal{L}_1$ by LLL lattice reduction algorithm with $\delta = \frac{3}{4}$. Let the reduced basis be $\mathbf{a_1}, ..., \mathbf{a_{t+1}}$, where $\mathbf{a_i} = (a_{i1}, ..., a_{it}, a_{it+1})$, $i = 1, ..., t+1$.
3. If $\|\mathbf{a_i}\| < 2^{\frac{\gamma}{t+1}}, i = 1, ..., t-z$, where z is a very small integer (relative to t), then solve the integer linear system with t unknowns $r_1, ..., r_t$ as follows

$$\sum_{j=1}^{t} a_{ij} \cdot r_i = \sum_{j=1}^{t} a_{ij} \cdot x_i, (i = 1, ..., t-z).$$

   Therefore, the integer solutions can be expressed as follow:

$$\mathbf{d} = \mathbf{d_0} + t_1 \mathbf{d_1} + ... + t_z \mathbf{d_z},$$

   where $\mathbf{d_0}$ is a special solution of the linear system, $t_1, ..., t_z$ are integers, $\mathbf{d_1}, ..., \mathbf{d_z}$ is a basis of integer solution space of the corresponding homogeneous linear equations.
4. Construct a lattice spanned by the row vectors $\mathbf{d_0}, \mathbf{d_1}, ..., \mathbf{d_z}$. Obviously, $(r_1, \ldots, r_t)$ is a short vector of the lattice. Thus we can find $r_1, \ldots, r_t$ by LLL algorithm.
5. Compute $p = \gcd(x_1 - r_1, x_2 - r_2)$. Return p.

Again, we note here that in our experiments $z \leq 2$, which is the key reason the algorithm works.

2.3 The relationship of t, γ, ρ

We could not use any theoretical analysis to tell us how to decide the right t, therefore we need to use computer experiments to help us to decide the right t, namely the smallest t such that we can get what we want. The experiments in this paper were carried out on two Quad-Core Intel Processor Q9400 CPUs (2.66 GHz) with 4 GB of main memory using MAGMA v.12-19. In one group of data, we fix $\eta = 1000$, and the running time and the relationship of t and γ, ρ are showed in Table 1.

γ	ρ	t	time(s)	γ	ρ	t	time(s)	γ	ρ	t	time(s)	γ	ρ	t	time(s)
5000	50	7	0.327	10000	50	23	24.382	15000	50	35	199.930	20000	50	45	846.820
5000	100	8	0.436	10000	100	23	24.180	15000	100	35	195.999	20000	100	45	859.643
5000	150	9	0.530	10000	150	23	24.975	15000	150	38	240.257	20000	150	48	912.371
5000	200	10	0.670	10000	200	28	38.142	15000	200	40	270.661	20000	200	50	1007.516
5000	250	11	0.826	10000	250	30	45.162	15000	250	42	312.610	20000	250	55	1296.196
5000	300	14	1.357	10000	300	30	45.770	15000	300	45	360.159	20000	300	62	1754.106
5000	350	16	1.794	10000	350	35	65.754	15000	350	46	378.240	20000	350	70	2419.840
5000	400	17	2.106	10000	400	36	70.855	15000	400	51	519.077	20000	400	75	2883.272
5000	450	18	2.386	10000	450	40	91.447	15000	450	59	749.179	20000	450	85	4245.879

Table 1: The relationship of t and γ, ρ

From the Table 1, we can observe that the relationship of t with γ and ρ is approximate linear. We did many more experiments, where γ is up to 200,000. With the large amount of data we have, using the least square method, we conclude that the data indicates that, if $N \in (2^{\gamma-1}, 2^{\gamma})$,

$$t \approx \lfloor 0.003566\gamma + 0.083526\rho - 30 \rfloor.$$

A much simpler observation is that

$$t \leq 0.005\gamma,$$

for all the data we collected. We will use this relation for our complexity estimate. Surely, the exactly relationship of t with γ, ρ is an open question.

We also did experiments for cases where $\rho \geq \eta/2$, the relationship of t with ρ and γ is much more complicated and t grows much faster.

2.4 The complexity of GACD algorithm

In the GACD algorithm, the dominant computation is the LLL reduction of our lattice and the rest can be neglected. Since, we need only to use one set of appropriate N and t once, the most complex calculations required of the GACD algorithm is one time LLL lattice reduction. From Theorem 3 , we know that the complexity of LLL lattice reduction algorithm is polynomial in γ and t for $\delta = 3/4$.

More specifically, let $\mathcal{L}$ be a lattice of rank $t+1$ with basis $\mathbf{b_1}, ..., \mathbf{b_{t+1}}$, and $\|\mathbf{b_i}\| \leq 2^{\gamma+1}, (i = 1, \ldots, t+1)$. Then the number of **bit operations** needed by the LLL basis reduction algorithm for $\delta = 3/4$ is

$$O\big((t+1)^6(\log_{4/3} 2^{\gamma+1})^3\big)$$

or

$$O\big((t+1)^6(2(\gamma+1))^3\big),$$

under school-multiplication, where $t \leq 0.005\gamma$.

Therefore, the number of **bit operations** needed by the GACD algorithm is $O\big((t+1)^6(2(\gamma+1))^3\big)$, under school-multiplication, where $t \leq 0.005\gamma$.

3 Cryptanalysis of Fully Homomorphic Encryption(FHE) Scheme Based on GACD Problem

In [16], the author build a FHE scheme over integer which based on the hardness of computing an approximate common divisor. The main appeal of this scheme is conceptual simplicity (compared to Gentry's[10]). In this section, we use our algorithm to attack the FHE scheme based on GACD Problem[16].

Let λ be a security parameter of the FHE scheme, γ be the bit-length of the integer in public key, η be the bit-length of the secret key, ρ be the bit-length of the noise, τ is the number of integers in the public key. The parameters generation goes as follows:
The secret key is an odd $\eta - bit$ integer:

$$p \leftarrow (2\mathbb{Z}+1)\bigcap[2^{\eta-1}, 2^{\eta}).$$

For the public key, sample $x_i \leftarrow \mathcal{D}_{\gamma,\rho}(p)$, for $i = 1, ..., \tau$ such that x_1 is the largest, where

$$\mathcal{D}_{\gamma,\rho}(p) = \{\text{choose } \ q \leftarrow \mathbb{Z}\bigcap[0, 2^{\eta}/p), r \leftarrow \mathbb{Z}\bigcap(-2^{\rho}, 2^{\rho}) : \text{output } \ x = pq + r\}.$$

The public key is $\text{pk} = (x_1, ..., x_\tau)$.

In [16], the authors proposed a convenient parameter set as:

$$\rho = \lambda, \eta = \mathcal{O}(\lambda^2), \gamma = \mathcal{O}(\lambda^5), \tau = \gamma + \lambda.$$

We would be able to solve the GACD problem with this parameter set in polynomial time of λ using our algorithm.

We consider the first t integers, where $t = \lfloor 0.003566\gamma + 0.083526\rho - 30 \rfloor$. Choose $N \in (2^{\gamma-1}, 2^{\gamma})$. Since $\rho < \frac{\eta}{2}$ in the parameter set , we can use our algorithm to find the secret p. The number of bit operations is about

$$O\big((0.005\lambda^5)^6(2(\gamma+1))^3\big)$$

or

$$O\big((0.005\lambda^5)^6(2(\lambda^5+1))^3\big)$$

We apply our algorithm to the parameters in [6] and we could break all the cases where their parameter $\gamma < 2^{20}$. We note here that the algorithm of Chen and Phone [4]relies only on ρ, while we are different, and we could break cases where $\rho \geq 60$ easily while their method can not. In addition, our algorithm in general requires small memory (roughly $O(0.005\gamma)^2 \times \gamma$ bits).

4 Conclusion and Discussion

In this paper, we present a new algorithm to solve the GACD problem. Through theoretical arguments and heuristically arguments based on experiments, we show that this algorithm can solve the GACD problem in polynomial time if the system parameters satisfies the relation $\rho < \eta/2$. This algorithm shows that there is a polynomial time algorithm to break the the fully homomorphic encryption schemes (FHE), which are based on the approximate common divisors problem, in terms of the system parameter λ.

It is remain an open problem to theoretical prove that indeed our algorithm works, in particular, why the short vectors from the LLL reduction satisfy the relation (1). One more interesting problem is to find out the exact complexity for our algorithm when $\eta/2 \leq \rho \leq \eta$.

References

1. Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE.Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on. IEEE, 2011: 97-106.
2. Brakerski Z. Fully homomorphic encryption without modulus switching from classical GapSVP.Advances in Cryptology-CRYPTO 2012. Springer Berlin Heidelberg, 2012: 868-886.

3. Bosma W, Cannon J, Playoust C. The Magma algebra system I: The user language. Journal of Symbolic Computation, 1997, 24(3): 235-265.
4. Chen Y, Nguyen P Q. Faster algorithms for approximate common divisors: Breaking fully homomorphic encryption challenges over the integers. Advances in Cryptology-EUROCRYPT 2012. Springer Berlin Heidelberg, 2012: 502-519.
5. Cohn H, Heninger N. Approximate common divisors via lattices. arXiv preprint arXiv:1108.2714, 2011.
6. Coron J S, Mandal A, Naccache D, et al. Fully homomorphic encryption over the integers with shorter public keys. Advances in Cryptology-CRYPTO 2011. Springer Berlin Heidelberg, 2011: 487-504.
7. Coron J S, Naccache D, Tibouchi M. Public key compression and modulus switching for fully homomorphic encryption over the integers. Advances in Cryptology-EUROCRYPT 2012. Springer Berlin Heidelberg, 2012: 446-464.
8. Cheon J H, Coron J S, Kim J, et al. Batch fully homomorphic encryption over the integers. Advances in Cryptology-EUROCRYPT 2013. Springer Berlin Heidelberg, 2013: 315-335.
9. Coppersmith D. Finding a small root of a univariate modular equation. Advances in Cryptology-EUROCRYPT96. Springer Berlin Heidelberg, 1996: 155-165.
10. Gentry C. A fully homomorphic encryption scheme. Stanford University, 2009.
11. Gentry C, Halevi S. Implementing gentry's fully-homomorphic encryption scheme. Advances in Cryptology-EUROCRYPT 2011. Springer Berlin Heidelberg, 2011: 129-148.
12. Howgrave-Graham N. Approximate integer common divisors. Cryptography and Lattices. Springer Berlin Heidelberg, 2001: 51-66.
13. Stehlé D, Steinfeld R. Faster fully homomorphic encryption. Advances in Cryptology-ASIACRYPT 2010. Springer Berlin Heidelberg, 2010: 377-394.
14. Micciancio D, Goldwasser S. Complexity of lattice problems: a cryptographic perspective. Springer, 2002.
15. Nguyen P Q, Valle B. The LLL algorithm: survey and applications. Springer Publishing Company, Incorporated, 2009.
16. Van Dijk M, Gentry C, Halevi S, et al. Fully homomorphic encryption over the integers. Advances in Cryptology-EUROCRYPT 2010. Springer Berlin Heidelberg, 2010: 24-43.
17. Stehlé D, Zimmermann P. A binary recursive gcd algorithm. Algorithmic number theory. Springer Berlin Heidelberg, 2004: 411-425.