A New Algorithm for Solving the General Approximate Common Divisors Problem and Cryptanalysis of the FHE Based on the GACD problem

Jintai Ding[1,2,3], Chengdong Tao[4]

1 University of Cincinnati, 2 ChongQing University, 3 Academia Sinica, 4 South China University of Technology

Abstract. In this paper, we propose a new algorithm for solving the general approximate common divisors (GACD) problems, which is based on lattice reduction algorithms on certain special lattices and linear equation solving algorithms over integers. Through both theoretical arguments and experimental data, we show that our new algorithm works in polynomial time but under roughly the following condition:

– There is a positive integer t such that

$$\frac{\gamma+\eta}{t}+\frac{t}{H}+\rho<\eta;$$

– We have more than t GACD samples.

or equivalently

–

$$H(\eta-\rho)^2-4(\gamma+\eta)>0$$

– We have more than $t=\lceil\frac{H(\eta-\rho)-\sqrt{H^2(\eta-\rho)^2-4H(\gamma+\eta)}}{2}\rceil$ GACD samples.

Here γ, η and ρ are parameters describing a GACD problem, $H=1/\log_2 F$ and F is the Hermite Factor. In our experiments, H is shown to be roughly 40 when using the LLL reduction algorithm and it should be around 80 when using Deep or BKZ algorithms. We show how our algorithm can be applied to attack the fully homomorphic encryption schemes which are based on the general approximate common divisors problem and its limitations.

Key words. General approximate common divisors problems; Fully homomorphic encryption; Lattice; LLL; BKZ

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 [14]. 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.*

When $r_1 = 0$, namely $x_1 = pq_1$, where q_1 is a very large integer without any small prime factors, this problem is called a partial approximate common divisors(PACD) problem.

In this paper, we will concentrate on the GACD problem, since the PACD problem is a special case of the GACD problem. Therefore, our algorithm can be used to solve PACD problems without any change. Furthermore, we believe we can improve our algorithm to solve the PACD problem in the case of applications to attack the FHE based on the PACD problems.

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 [20]. At CRYPTO'11, Coron et al. presented a more efficient variant of the FHE scheme in [7] 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[22]. 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 futile.

In [14], 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 [6], 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 [7] is actually slower than exhaustive search on the challenges, and therefore much slower than the attack in [4].

The contribution of this paper: 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 [6], 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 the LLL algorithm. Then we can recover p via Euclidean algorithm.

The algorithm can be summarized as follows:

1. We consider the first $t = \lceil \frac{H(\eta-\rho)-\sqrt{H^2(\eta-\rho)^2-4H(\gamma+\eta)}}{2} \rceil$ integers $x_1, ..., x_{t-1}$ and construct a lattice $\mathcal{L}$ spanned by rows of the following matrix:

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

   Here N is a random number of $\gamma + \eta$ bits.
2. We apply a lattice reduction algorithm to find a short vector and we can prove that the shortest vector $(u_1, ...u_{t-1}, v_t)$ gives solution to the equation:

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

   which implies that $\sum_{i=1}^{t-1} u_i \cdot q_i = 0$. In our experiments, we use the LLL lattice reduction algorithm with $\delta = \frac{3}{4}$.
3. Theoretically, we need to choose $t-2$ (or a few more) different N, such that we will get $t-2$ such equations above. In practice, the LLL reduction in general actually gives us $t-2$ such vectors, each vector gives us a linear equation satisfied by $r_1, \ldots, r_{t-1}$. We find the integer solutions of this equations by solving the derived linear system in integers. The integer solutions can be expressed as follow:

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

   where $\mathbf{d_0}$ is a special solution of the linear system, t_1 is integer, $\mathbf{d_1}$ is a basis of integer solution space of the corresponding homogeneous linear equations.
4. We construct a lattice spanned by the row vectors $\mathbf{d_0}, \mathbf{d_1}$. Obviously, $(r_1, \ldots, r_{t-1})$ is a short vector of the lattice. Thus we can find $r_1, \ldots, r_{t-1}$ by the LLL algorithm.
5. Finally, we recover the common divisor p by Euclidean algorithm.

Our algorithm works under the assumption that such a t exists, namely

$$H^2(\eta-\rho)^2 - 4H(\gamma+\eta)^2 \geq 0,$$

.

It is clear that $t < H(\eta - \rho)$, and that our method is polynomial time in terms of γ, η, ρ if we use a polynomial time lattice reduction algorithm, since the main time consumption step is the lattice reduction step, in particular, if we use the LLL lattice reduction algorithm with $\delta = \frac{3}{4}$.

If such a t does not exist, our algorithm will fail because we can not promise the shortest vector from the LLL reduction would give us a vector that provides a solution to build the linear equations satisfied by the error term r_i. This is why when we apply our algorithm to attack the GACD problem for the FHE system by Van Dijk et al. [20], where $\gamma = \lambda^5, \eta = \lambda^2, \rho = \lambda$, we would fail in general, since, for a relatively large λ, we can not find such a t, but we show if we make any real progress in terms of improving the Hermit factor, the situation can be very different and we will discuss how our algorithm could significantly

impact the security of these FHEs. Thus a significant contribution of ours is that we show that the Hermite factor F (or more directly the constant $H = 1/\log_2 F$) could significantly impact the selection of the security parameters of these FHE systems.

Organization: In Section 3, we present our main results and we conclude the paper in Section 4.

2 Main Results

We present some facts on lattice and more details can be found in [16] and [17].

2.1 Preliminaries

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

Definition 2.1. *[16]. 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 2.2. *[16]. 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 2.3. *[16]. 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 2.4. *[17]. 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.

Theorem 2.5. *[17]. 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. $\prod_{i=1}^{n} \|\mathbf{a_i}\| \leq 2^{\frac{n(n-1)}{4}} det(\mathcal{L})$.

We note here that the first estimate here is not optimal in terms of practice. The real Hermite factor is much smaller, which is very critical to our algorithm.

2.2 The new GACD algorithm

We describe our algorithm for the GACD problem. We consider the first $t-1$ integers $x_1, \ldots, x_{t-1}$. The algorithm is defined as follow:

Algorithm 1 .
Input: Samples $x_1, \ldots, x_{t-1}$.
Output: Integer p .

1. We randomly select an integer of $\gamma + \eta$ bits. We build the matrix L as:

$$L = \begin{pmatrix} 1 & & & x_1 \\ & 1 & & x_2 \\ & & \ddots & \vdots \\ & & & 1 \; x_{t-1} \\ & & & N \end{pmatrix}.$$

2. $\mathbf{a_1}, ..., \mathbf{a_t} \leftarrow LLLBasis(L)$, where $\mathbf{a_i} = (a_{i1}, ..., a_{it})$, $(i = 1, ..., t)$, $\|\mathbf{a_1}\| < ... < \|\mathbf{a_t}\|$.
3. We repeat Step 1 and Step 2 for $t-2$ times each with different N, for each round (the j-th), we keep the shortest vector $\mathbf{a_1}$ and call it $\mathbf{b_j}$.
4. Solve the integer linear system with $t-1$ unknowns $r_1, ..., r_{t-1}$ as follows

$$\sum_{j=1}^{t-1} b_{ij} \cdot r_i = \sum_{j=1}^{t-1} b_{ij} \cdot x_i, (i = 1, ..., t-2).$$

   The integer solutions can be expressed as follow:

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

   where $\mathbf{d_0}$ is a special solution of the linear system, t_1 is integer, $\mathbf{d_1}$ is a basis of integer solution space of the corresponding homogeneous linear equations.
5. Construct a lattice spanned by the row vectors $\mathbf{d_0}, \mathbf{d_1}$. Obviously, $(r_1, \ldots, r_{t-1})$ is a short vector of the lattice. Thus we can find $r_1, \ldots, r_{t-1}$ by LLL algorithm.
6. $p = \gcd(x_1 - r_1, ..., x_{t-1} - r_{t-1})$. Return p.

To prove the algorithm indeed works, we need a more precise condition on t, which is

$$\frac{\gamma + \eta}{t} + \frac{t}{H} + \rho + \log_2 \sqrt{t-1} + 1 < \eta.$$

Since t is small compared to the rest, this condition is essentially the same as the one we have in the abstract.

We start from the first observation that is needed for explaining our algorithm.

Lemma 2.6. *Let $u_1, ..., u_{t-1}$ be integers, then $\sum_{i=1}^{t-1} u_i x_i = \sum_{i=1}^{t-1} u_i r_i$ if and only if $\sum_{i=1}^{t-1} u_i q_i = 0$.*

Proof. It is evident.

We construct a lattice $\mathcal{L}$ spanned by rows of the matrix:

$$L = \begin{pmatrix} 1 & & & x_1 \\ & 1 & & x_2 \\ & & \ddots & \vdots \\ & & & 1 \; x_{t-1} \\ & & & N \end{pmatrix}. \tag{2}$$

Let $\mathbf{v} \in \mathcal{L}$, then $\mathbf{v}$ has the form

$$\mathbf{v} = (u_1, \ldots, u_{t-1}, \sum_{i=1}^{t-1} u_i \cdot x_i + u_t N),$$

where $u_1, \ldots, u_t$ are integers. Thus the length of vector $\mathbf{v}$ in Euclid norm is

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

Lemma 2.7. *For any vector $\mathbf{v} \in \mathcal{L}$ where t satisfies the condition above, if $\|\mathbf{v}\| < 2^{\eta-\rho-1-\log_2 \sqrt{t-1}}$, then*

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

Proof. let $\mathbf{u} = (u_1, ..., u_{t-1})$ and $\mathbf{x} = (x_1, ..., x_{t-1})$. Since each $x_i (i = 1, ..., t-1)$ is of γ bits, this implies that

$$|\sum_{i=1}^{t-1} u_i \cdot x_i| = |\mathbf{u} \cdot \mathbf{x}| \leq \|\mathbf{u}\| \|\mathbf{x}\| |\cos(\theta)| \leq 2^{\gamma+1} \sqrt{t-1} \|\mathbf{u}\| \leq 2^{\gamma+1} \sqrt{t-1} \|\mathbf{v}\|.$$

Here θ is the angle between $\mathbf{u}$ and $\mathbf{x}$.

Due to $\|\mathbf{v}\| < 2^{\eta-\rho-\log_2 \sqrt{t-1}-1}$, this implies that

$$|\sum_{i=1}^{t-1} u_i \cdot x_i| < 2^{\gamma+\eta-\rho} < 2^{\gamma+\eta-1} \leq N/2.$$

Therefore this implies there is no modular N operation involved and

$$u_t = 0.$$

Thus we have

$$\mathbf{v} = (u_1, \ldots, u_{t-1}, \sum_{i=1}^{t-1} u_i \cdot x_i),$$

This leads to

$$\sum_{i=1}^{t-1} u_i \cdot x_i + u_t N = \sum_{i=1}^{t-1} u_i \cdot x_i = p \sum_{i=1}^{t-1} u_i \cdot q_i + \sum_{i=1}^{t-1} u_i \cdot r_i.$$

If

$$\sum_{i=1}^{t-1} u_i \cdot q_i \neq 0,$$

we have

$$p|\sum_{i=1}^{t-1} u_i \cdot q_i| > 2^{\eta}.$$

Since r_i is smaller than 2^{ρ}, this implies that

$$|\sum_{i=1}^{t-1} u_i \cdot r_i| \leq 2^{\rho}\sqrt{t-1}\|\mathbf{v}\| < 2^{\eta-1}.$$

This implies

$$|\sum_{i=1}^{t-1} u_i \cdot x_i| = |p\sum_{i=1}^{t-1} u_i \cdot q_i + \sum_{i=1}^{t-1} u_i \cdot r_i| \geq 2^{\eta-1}.$$

This implies that it is impossible to have that

$$\sum_{i=1}^{t-1} u_i \cdot q_i \neq 0.$$

Therefore

$$\sum_{i=1}^{t-1} u_i \cdot q_i = 0.$$

This finishes the proof.

This implies that

Theorem 2.8. *If* $\frac{\gamma+\eta}{t} + \frac{t}{H} + \rho + \log_2\sqrt{t-1} + 1 < \eta$, *for the shortest vector* $\mathbf{v}$ *we derive from the lattice reduction on* L, *we also have*

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

Proof. We use the lattice reduction algorithm to derive a short vector $\mathbf{v}$ whose length is less than or equal to

$$F^t det(L)^{1/t} = 2^{\frac{\gamma+\eta}{t}+\frac{t}{H}},$$

which implies that the length of the vector $\mathbf{v} = (u_1, ..., u_{t-1}, \sum_{i=1}^{t-1} u_i x_i)$ is also less than $2^{\frac{\gamma+\eta}{t}+\frac{t}{H}}$:

$$\|\mathbf{v}\| \leq 2^{\frac{\gamma+\eta}{t}+\frac{t}{H}} < 2^{\eta-\rho-1-\log_2\sqrt{t-1}},$$

which is due to the condition $\frac{\gamma+\eta}{t} + \frac{t}{H} + \rho + \log_2\sqrt{t-1} + 1 < \eta$. From the lemma above, we finish the proof.

Theorem 2.9. *The Algorithm 1 succeeds with high probability.*

Proof. First by counting, we can show that in the range of the same length of the shortest vector of the reduction, there are a large of vectors satisfying the condition (1). Also we know that when we choose different N, we would derive different short vectors which also satisfies the condition (1). It is very reasonable to assume that each time we derive a random vector which satisfies the condition (1), and a set of $t-2$ of them should have a very high probability to be linearly independent, which promises that our algorithm would succeed.

In our experiments, we use the LLL reduction with $\delta = 3/4$, and we find out that H should be selected as 40. Also a very interesting situation is that after the LLL reduction, the first $t-2$ vectors are all of the around the same length, which implies that we will get $t-2$ equation satisfies by $r_i (i = 1, ..., t-1)$, which is why our algorithm is successful in general with only one round of operation of Step 1 and Step 2.

From the the practical point of view, the condition

$$\frac{\gamma+\eta}{t} + \frac{t}{H} + \rho + \log_2 \sqrt{t-1} + 1 < \eta,$$

looks too complicated and since t is very small comparatively, we prefer to simplified it as:

$$\frac{\gamma+\eta}{t} + \frac{t}{H} + \rho < \eta,$$

which is much easier to handle as explained in the abstract.

In all the experiments we have done, we have not yet found any case, it failed.

From all the above, we conclude that we can give a polynomial time algorithm to solve the GACD problem as long as we can find a t satisfying the condition we give. Surely not all parameters satisfies the condition.

2.3 Complexity analysis of Algorithm 1

Since the most complex calculations required of the Algorithm 1 is the lattice reduction algorithm. In [21], Novocin et al. proposed a LLL-reduction algorithm with quasi-linear time complexity in term of number of input bits, which is very useful for our algorithm since it involves matrices with large entries. This algorithm was as effective as LLL reduction with $\delta = \frac{3}{4}$.

Let $\mathcal{L}$ be a lattice of rank t with basis $\mathbf{b_1}, ..., \mathbf{b_t}$, and $\|\mathbf{b_i}\| \leq 2^{\gamma+1}, (i = 1, \ldots, t)$. It takes as input an arbitrary lattice basis L; It computes a basis of L which is reduced for a mild modification of the Lenstra-Lenstra-Lovász reduction; It terminates in time

$$O(t^{5+\varepsilon}(\gamma+1) + t^{\omega+\varepsilon+1}(\gamma+1)^{1+\varepsilon}),$$

where $\varepsilon > 0$ and ω is a valid exponent for matrix multiplication. This algorithm with a time complexity is quasi-linear in γ and polynomial in t.

Therefore, the number of bit operations needed by the Algorithm 1 is roughly

$$O(t^{5+\varepsilon}(\gamma+1) + t^{\omega+\varepsilon+1}(\gamma+1)^{1+\varepsilon}),$$

under school-multiplication, where

$$t = \lceil \frac{H(\eta-\rho) - \sqrt{H^2(\eta-\rho)^2 - 4H(\gamma+\eta)}}{2} \rceil.$$

2.4 A more practical variant and experiments

Later, we modify algorithm a little bit to make things easier. The algorithm can be summarized as follows:

1. We consider the first $t = \lceil \frac{H(\eta-\rho)-\sqrt{H^2(\eta-\rho)^2-4H(\gamma+\eta)}}{2} \rceil$ integers $x_1, ..., x_t$ and construct a lattice $\mathcal{L}$ spanned by rows of the following matrix:

$$L = \begin{pmatrix} 1 & & & x_1 \\ & 1 & & x_2 \\ & & \ddots & \vdots \\ & & & 1 \; x_{t-1} \\ & & & x_t \end{pmatrix}.$$

2. We apply LLL lattice reduction algorithm with $\delta = \frac{3}{4}$ to find a short vector and it turns that this short vector gives solution to the equation:

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

   which implies that $\sum_{i=1}^{t} u_i \cdot q_i = 0$.
3. The LLL reduction in general gives us $t-1$ such vectors, 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:

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

   where $\mathbf{d_0}$ is a special solution of the linear system, t_1 is integer, $\mathbf{d_1}$ is a basis of integer solution space of the corresponding homogeneous linear equations.
4. We construct a lattice spanned by the row vectors $\mathbf{d_0}, \mathbf{d_1}$. 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. Finally, we recover the common divisor p by Euclidean algorithm.

We call this algorithm the Algorithm 2.

This algorithm works as well as the Algorithm 1, and most of our experiments are done using this algorithm.

Note here that unlike the case of the Algorithm 1, we can not prove that this algorithm actually works.

We carry out many experiments using Magma 2.18-9 about various parameters which satisfy $H(\eta-\rho)^2 - 4(\gamma+\eta) > 0$. We run all the experiments on a Sun X4440, with four Quad-Core AMD OpteronTM Processor 8356 CPUs and 128 GB of main memory. Each CPU is running at 2.3 GHz. W can not list all the details of the experiments but some of them and they are contained in the tables below.

λ	10	15	20	25	30	35	40	50
t	13	18	23	28	33	38	43	53
time(s)	0.03	0.16	0.8	3.22	11.930	38.27	120.59	754.89

Table 1: The running times and the dimension of lattice using algorithm 2 to successfully recover p on the parameters $\rho = \lambda, \eta = \lambda^2, \gamma = \lambda^3$.

λ	8	9	10	11	12
t	77	96	116	139	164
time(s)	23.98	103.35	270.76	1495.98	7062.94

Table 2: The running times and the dimension of lattice using algorithm 2 to successfully recover p on the parameters $\rho = \lambda, \eta = \lambda^2, \gamma = \lambda^4$.

In our experiments, the constant H is roughly 40.

2.5 Application of our algorithm to attack the FHE

The GACD problem was used for constructing the FHE schemes. For the FHE, the choice of parameters γ, η, ρ are selected from two direction, the first one is to make the system works, the second one is to make sure the system is as efficient as possible and the second comes from security concerns.

First, it is very clear that if we select

$$\gamma = O(\lambda^4); \eta = O(\lambda^2); \rho = \lambda,$$

where λ is security parameter. Our algorithm solves all the cases in polynomial time in general.

Second it is also very clear that if we use the parameter suggested by Van Dijk et al. [20]:

$$\gamma = O(\lambda^5); \eta = O(\lambda^2); \rho = \lambda;$$

our algorithm would fail asymptotically once λ is big enough. But practically things becomes more subtle.

If we choose

$$\gamma = \lambda^5; \eta = \lambda^2; \rho = \lambda;$$

then, if $H = 40$, our algorithm would succeed till

$$\lambda = 8;$$

while if $H = 80$, for example, for Deep or BKZ reduction algorithm, our algorithm would succeed till

$$\lambda = 18,$$

and if we push the reduction algorithm to be more effective, for instance as suggested [13], where

$$F = 1.005, H = 138,$$

our algorithm would work up to

$$\lambda = 33.$$

This is clear that our algorithm fails completely.

If one wants to more aggressive to choose

$$\gamma = \lambda^5/10; \eta = \lambda^2; \rho = \lambda;$$

then, if $H = 40$, our algorithm would succeed till

$$\lambda = 98;$$

while if $H = 80$, for example, for Deep or BKZ reduction algorithm, our algorithm would succeed till

$$\lambda = 198,$$

and $H = 138$, our algorithm would work up to

$$\lambda = 343.$$

In this case, our algorithm works pretty well.

Furthermore, if one wants to even more aggressive to choose

$$\gamma = \lambda^5/20; \eta = \lambda^2; \rho = \lambda;$$

then, if $H = 40$, our algorithm would succeed till

$$\lambda = 198;$$

while if $H = 80$, for example, for Deep or BKZ reduction algorithm, our algorithm would fail at

$$\lambda = 398,$$

and if $H = 138$ our algorithm would work up to

$$\lambda = 688.$$

In this case, our algorithm would practically be a polynomial time algorithm to attack the corresponding FHEs.

One may ask about the efficiency of our algorithm.

For the case,

$$\gamma = \lambda^5/10; \eta = \lambda^2; \rho = \lambda;$$

if we choose $\lambda = 80$ and have $H = 40$, then we need to have $t = 72833$, then the complexity of our algorithm is 2^{109} according to [21]. In this case, the complexity of attack in [4] would be 2^{148}.

For the case,

$$\gamma = \lambda^5/20; \eta = \lambda^2; \rho = \lambda;$$

if we choose $\lambda = 80$ and have $H = 40$, then we need to have $t = 29328$, then the complexity of our algorithm is 2^{101} according to [21]. In this case, the complexity of attack in [4] would be 2^{147}.

For the case,

$$\gamma = \lambda^5/20; \eta = \lambda^2; \rho = \lambda;$$

if we choose $\lambda = 120$ and have $H = 40$, then we need to have $t = 107274$, then the complexity of our algorithm is 2^{113} according to [21]. In this case, the complexity of attack in [4] would be 2^{210}.

For the case,

$$\gamma = \lambda^5/20; \eta = \lambda^2; \rho = \lambda;$$

if we choose $\lambda = 160$ and have $H = 40$, then we need to have $t = 287077$, then the complexity of our algorithm is 2^{122} according to [21]. In this case, the complexity of attack in [4] would be 2^{272}.

This clearly shows the advantage of our algorithm compared to the best ones before.

3 Conclusion

We present a new lattice reduction based algorithm to solve the GACD problem. This algorithm works in polynomial time if the parameter satisfies certain condition on the proper dimension of lattice we need. This restrict is vert much related to the Hermit factor of the reduction algorithm. We show that we can use our algorithm to attack the FHE systems based on the GACD where $\gamma = O(\lambda^5), \eta = O(\lambda^2), \rho = \lambda$. and how the Hermite factor can impact the selection of secure parameters.

From the theoretical point of view, our work is the first one that related the selection of parameters directly related to the Hermite parameters, which was not known before, More importantly, we show that if we make progress on improve the Hermit factors, the security parameters might be fundamentally affected.

For future work, we think our attack point to a very interesting new direction, which has great potential in term of further improvement in both theory and practice, and we believe we can further improve our algorithm to solve the PACD problem.

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. Chen Y, Nguyen P Q. BKZ 2.0: better lattice security estimates. Advances in CryptologyCASI-ACRYPT 2011. Springer Berlin Heidelberg, 2011: 1-20.
6. Cohn H, Heninger N. Approximate common divisors via lattices. arXiv preprint arXiv:1108.2714, 2011.
7. 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.
8. 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.
9. 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.
10. Coppersmith D. Finding a small root of a univariate modular equation. Advances in Cryptology-EUROCRYPT96. Springer Berlin Heidelberg, 1996: 155-165.
11. Gentry C. A fully homomorphic encryption scheme. Stanford University, 2009.
12. Gentry C, Halevi S. Implementing gentrys fully-homomorphic encryption scheme. Advances in Cryptology-EUROCRYPT 2011. Springer Berlin Heidelberg, 2011: 129-148.
13. Gama N, Nguyen P Q. Predicting lattice reduction. Advances in CryptologyCEUROCRYPT 2008. Springer Berlin Heidelberg, 2008: 31-51.
14. Howgrave-Graham N. Approximate integer common divisors. Cryptography and Lattices. Springer Berlin Heidelberg, 2001: 51-66.
15. Stehlé D, Steinfeld R. Faster fully homomorphic encryption.Advances in Cryptology-ASIACRYPT 2010. Springer Berlin Heidelberg, 2010: 377-394.

16. Micciancio D, Goldwasser S. Complexity of lattice problems: a cryptographic perspective. Springer, 2002.
17. Nguyen P Q, Valle B. The LLL algorithm: survey and applications. Springer Publishing Company, Incorporated, 2009.
18. Nguyen P Q, Stehlé D. LLL on the average.Algorithmic Number Theory. Springer Berlin Heidelberg, 2006: 238-256.
19. Nguyen P Q, Stehlé D. An LLL algorithm with quadratic complexity. SIAM Journal on Computing, 2009, 39(3): 874-903.
20. 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.
21. Novocin A, Stehlé D, Villard G. An LLL-reduction algorithm with quasi-linear time complexity. Proceedings of the 43rd annual ACM symposium on Theory of computing. ACM, 2011: 403-412.
22. Stehlé D, Zimmermann P. A binary recursive gcd algorithm. Algorithmic number theory. Springer Berlin Heidelberg, 2004: 411-425.

Appendix: The Magma program of algorithm 2

```
clear;
Z := IntegerRing();
function SampleGen(rho,eta,gamma,tau)
        theta := gamma - eta;
        x := [];
        q := [];
        r := [];
        p := Random(2^eta);
        q[1] := Random(2^theta);
        r[1] := Random(-2^rho, 2^rho);
        x[1] := q[1]*p + r[1];
        for i in [2..tau] do
                q[i] := Random(2^theta);
                r[i] := Random(-2^rho, 2^rho);
                x[i] := q[i] * p + r[i];
        end for;
        return x,p,q,r;
end function;

function Gacd(x,rho,eta,gamma,H)
        Delta := Z!Floor(Sqrt((H^2 * (eta - rho)^2) - 4 * H * (gamma + eta)));
        t := Z!Ceiling((H * (eta - rho) - Delta)/2) + 10;
        B := ZeroMatrix(Z,t,t);
        for i in [1..t-1] do
                B[i,t] := x[i] ;
                B[i,i] := 1;
        end for;
        B[t,t] := x[t];
        SetVerbose("LLL", 1);
        L := Lattice(B);
        L := LLLBasisMatrix(L);
        row := Nrows(L);
        col := Ncols(L);
        C := ZeroMatrix(Z,row-1,col);
        for i in [1..row-1] do
                temp := 0;
                for j in [1..col-1] do
                        C[i,j] := L[i,j];
                        temp +:= C[i][j]*x[j]
                end for;
                C[i][col] := (L[i][col]-temp) div x[t];
        end for;
        v := [Z!L[i][col]: i in [1..row-1]];
        v := Vector(v);
        recover,bs := Solution(Transpose(C), v);
        rb := ElementToSequence(recover);
        db := Dimension(bs);
```

```
        D := [];
        for i in [1..db] do
                D[i] := ElementToSequence(bs.i);
        end for;
        MA := ZeroMatrix(Z,col,db+1);
        for i in [1..col] do
                for j in [1..db] do
                        MA[i,j] := D[j,i];
                end for;
                MA[i,db+1] := rb[i];
        end for;
        LM := Lattice(Transpose(MA));
        re := ShortestVectorsMatrix(LM);
        seq := [x[i] - re[1][i]: i in [1..col]];
        seqplu := [x[i] + re[1][i]: i in [1..col]];
        pa := GCD(seq);
        pb := GCD(seqplu);
        return pa,pb;
end function;

lambda := 10;
rho := lambda;
eta := Round(lambda^2);
gamma := Round(lambda^4);
tau := 1000;
H := 40;
SetOutputFile("gacdtest.txt":Overwrite:=true);
x,p,q,r := SampleGen(rho,eta,gamma,tau);
pa,pb := Gacd(x,rho,eta,gamma,H);
print"revover p:= " ,pa;
print"revover p:= " ,pb;
UnsetOutputFile();
```