Algorithms for the Approximate Common Divisor Problem

Steven D. Galbraith[1], Shishay W. Gebregiyorgis[1], and Sean Murphy[2]

[1] Mathematics Department,
University of Auckland,
New Zealand.
S.Galbraith@math.auckland.ac.nz,sgeb522@aucklanduni.ac.nz
[2] Royal Holloway, University of London, UK.
S.Murphy@rhul.ac.uk

Abstract. The security of homomorphic encryption over the integers and its variants depends on the hardness of the Approximate Common Divisor (ACD) problem. In this paper we review and compare existing algorithms to solve the ACD problem using lattices. In particular we consider the simultaneous Diophantine approximation method, the orthogonal lattice method, and a method based on multivariate polynomials and Coppersmith's algorithm that was studied in detail by Cohn and Heninger. We give a novel analysis of these algorithms that is appropriate for some of the recent variants of the ACD problem.
One of our main contributions is to compare the multivariate polynomial approach with other methods. We find that Cohn and Heninger made certain assumptions that give a misleading view of the best choices of parameters for that algorithm. Instead, the best parameters seem to be those for which the algorithm becomes the orthogonal lattice algorithm.
Another contribution is to consider a sample-amplification technique for ACD samples, and to consider a pre-processing algorithm similar to the Blum-Kalai-Wasserman (BKW) algorithm for learning parity with noise. We explain why, unlike in other settings, the BKW algorithm does not give an improvement over the lattice algorithms.
Keywords: Approximate common divisors, lattice attacks, orthogonal lattice, Coppersmith's method.

1 Introduction

The approximate common divisor problem (ACD) was first studied by Howgrave-Graham [HG01]. Further interest in this problem was provided by the homomorphic encryption scheme of van Dijk, Gentry, Halevi and Vaikuntanathan [DGHV10] and its variants [CMNT11,CNT12,CS15]. The computational problem is to determine a secret integer p when one is given many samples of the form $x_i = pq_i + r_i$ for small error terms r_i. More precisely, p is an η bit odd prime, the x_i are γ bits, and the r_i are ρ bits, where ρ is significantly smaller than η.

The original papers [HG01,DGHV10] sketched a large number of possible lattice attacks on this problem. Futher cryptanalytic work was done by [CN12,CNT12,CH13,DT14]. The main aim of our paper is to compare all known lattice attacks on the approximate

common divisor problem. Rather than determining the exact running time of these attacks, the main focus in [DGHV10] was to determine parameters for which the attacks do not work, and so the analysis was not very precise. Cohn and Heninger [CH13] analysed a method based on multivariate polynomials, but did not compare it with the orthogonal lattice methods in [DGHV10], and also their analysis was focussed on the case where only a small number of ACD samples are available. In contrast, we study these algorithms in the cryptographically relevant situation where the number of ACD samples is very large. We also consider these methods in the context of more recent variants of the ACD problem, such as by Cheon and Stehlé [CS15]. Hence, our analysis of these algorithms with respect to a common set of lattice reduction heuristics is a significant contribution to the literature on the problem. One of our main conclusions is that the multivariate polynomial approach is not better than the orthogonal lattice approach. We also propose a pre-processing idea, motivated by the Blum-Kalai-Wasserman algorithm for learning parity with noise (LPN), and a sample-amplification idea motivated by work on LPN and learning with errors (LWE).

We do not consider in this paper the variants of "exhaustive search" over the errors r_i, as proposed by Chen and Nguyen [CN12], Coron, Naccache and Tibouchi [CNT12], and Lee and Seo [LS14]. Such algorithms are important for the original version of the ACD problem, but are less relevant for the Cheon-Stehlé variant.

2 Background and Notation

We use standard notation throughout the paper. The symbols $\ll$ and $\gg$ do not have a precise technical meaning, but are supposed to convey an informal assurance of "significantly less (greater) than".

2.1 Statement of the approximate common divisor problems

There are at least four variants of the approximate common divisor problem in the literature. We now define these problems precisely.

Fix $\gamma, \eta, \rho \in \mathbb{N}$. Let p be an η-bit odd prime. By this we mean that

$$2^{\eta-1} < p < 2^{\eta}.$$

Actually it is not necessary for p to be prime, and in some applications (e.g., Appendix D of [DGHV10]) it is definitely not prime. Define the efficiently sampleable distribution $\mathcal{D}_{\gamma,\rho}(p)$ as

$$\mathcal{D}_{\gamma,\rho}(p) = \{pq + r \mid q \leftarrow \mathbb{Z} \cap [0, 2^{\gamma}/p), r \leftarrow \mathbb{Z} \cap (-2^{\rho}, 2^{\rho})\}. \quad (1)$$

In practice we have ρ significantly smaller than η and so all samples from $\mathcal{D}_{\gamma,\rho}(p)$ will satisfy $x_i < 2^{\gamma}$ with overwhelming probability. Note also that if t is sufficiently large and $x_1, \ldots, x_t$ are sampled from $\mathcal{D}_{\gamma,\rho}(p)$ then we expect there to be at least one index i such that

$$2^{\gamma-1} < x_i < 2^{\gamma}.$$

Definition 1. *Let notation be as above. The **approximate common divisor problem** (ACD) is: Given polynomially many samples x_i from $\mathcal{D}_{\gamma,\rho}(p)$, to compute p.*

*The **partial approximate common divisor problem** (PACD) is: Given polynomially many samples x_i from $\mathcal{D}_{\gamma,\rho}(p)$ and also a sample $x_0 = pq_0$ for uniformly chosen $q_0 \in \mathbb{Z} \cap [0, 2^\gamma/p)$, to compute p.*

In this paper we focus on the "computational" versions of the problems. There are also "decisional" versions, but it is known that the computational and decisional problems are equivalent. Furthermore, there are no known lattice attacks that directly solve a decisional problem without first essentially solving the computational problem.

Let λ be a security parameter. Van Dijk et al [DGHV10] set $\gamma/\eta^2 = \omega(\log(\lambda))$ to thwart lattice attacks on the approximate common divisor problem. Concretely, the parameters are set to $(\rho, \eta, \gamma) = (\lambda, \lambda^2, \lambda^5)$, so one sees that ρ is extremely small compared with η. The analysis in [DGHV10] is very conservative and seems to overestimate the size of γ required. For example, in [CNT12] one finds parameters $(\rho, \eta, \gamma) = (71, 2698, 19350000)$ that are claimed to have security level around 72-bits, and it is likely that γ can be taken considerably smaller than this while still achieving the claimed security level.

Cheon et al [CCK+13] have given a homomorphic encryption scheme that uses the Chinese remainder theorem to pack more information into a ciphertext. This system features ℓ η-bit primes p_i. Let $\pi = p_1 \cdots p_\ell$ and $x_0 = \pi q_0$. A ciphertext is an element $c = \pi q + r$ where r is congruent modulo each prime p_i to a small integer r_i, and information can be encoded in each value r_i (these are called CRT-components). The public key includes a number of ciphertexts x_i that are encryptions of 0, as well as a number of ciphertexts that are encryptions of 1 in a single CRT component. We refer to [CCK+13] and Chapter 7 of Lepoint [Lep14] for more details about parameters. We call the problem of computing $p_1, \ldots, p_\ell$ from the public key the **CRT-ACD problem**.

An important detail about CRT-ACD is that, since π is very large compared with an individual p_i, one can use smaller values for the q. In terms of cryptanalysis, the problem can be reduced to a standard PACD instance of the form $x_0 = p_1 q_0'$ and $x_i = p_1 q_i' + r_i'$, and it is these attacks that are used to specify the parameters. A reduction is given in Lemma 1 of [CCK+13] that gives evidence that the CRT variant of the ACD problem is hard, but this reduction does not preserve the sizes of parameters and so it is not very useful for setting concrete parameters. It is an open problem to give an algorithm to solve the CRT-ACD problem that exploits the CRT structure.

Cheon and Stehlé [CS15] have given a scale-invariant homomorphic encryption scheme that permits a very different flavour of parameters. Furthermore, they give an explicit hardness result for their parameters, by showing that if one can solve the (decisional) approximate common divisor problem then one can solve the (decisional) learning with errors problem. The parameters in [CS15] are set as

$$(\rho, \eta, \gamma) = (\lambda, \lambda + d\log(\lambda), \Omega(d^2 \lambda \log(\lambda))),$$

where d is the depth of the circuit to be evaluated homomorphically. Note that ρ is no longer extremely small compared with η. We will sometimes refer to these parameters as the **Cheon-Stehlé approximate common divisor problem**. We draw the reader's

attention to a typo in [CS15]: regarding the security of the parameters against the multivariate polynomial attack the authors wrote $\gamma < \eta^2$ but should have written $\gamma > \eta^2$; in any case the condition $\gamma > \eta^2$ is not required to have secure parameters.

We will see that the lattice algorithms for ACD seem to work less well for the CRT-ACD and Cheon-Stehlé-ACD. Hence these two variants seem to offer a higher degree of security, at least according to our current knowledge. This is perhaps not surprising in the case of the Cheon-Stehlé-ACD, since that problem enjoys some evidence for its hardness.

2.2 Lattice basis reduction

The algorithms considered in this paper make use of lattice basis reduction algorithms such as LLL [LLL82]. Recall that a lattice of rank n is a discrete subgroup of $\mathbb{R}^m$ that has rank n as a $\mathbb{Z}$-module. In this paper we write elements of a lattice as row vectors. Denote by $\langle \mathbf{u}, \mathbf{v} \rangle$ the Euclidean inner product on $\mathbb{R}^m$ and $\|\mathbf{v}\| = \langle \mathbf{v}, \mathbf{v} \rangle^{1/2}$ the Euclidean norm. We sometimes use the norm $\|(v_1, \ldots, v_n)\|_1 = \max\{|v_i|\}$. A lattice L is described by giving n basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$, such that $L = \{\sum_{i=1}^n a_i \mathbf{v}_i : a_i \in \mathbb{Z}\}$.

The volume of a lattice L, denoted $\det(L)$, is the volume of the paralleliped formed by any basis of the lattice. The successive minima $\lambda_i(L)$ are the smallest real numbers such that L contains i linearly independent vectors all of Euclidean norm less than or equal to $\lambda_i(L)$. So $\lambda_1(L)$ is the length of the shortest non-zero vector in the lattice L. The Gaussian heuristic states that, for a "random lattice", the shortest non-zero vector in the lattice has Euclidean norm approximately $\sqrt{n/(2\pi e)}\det(L)^{1/n}$. For details of the Gaussian heuristic see Ajtai [Ajt06] (formalising what is meant by a "random lattice" is non-trivial and is beyond the scope of this paper). A commonly used heuristic is that if L is a lattice that contains a vector $\mathbf{v}$ of Euclidean norm less than $\det(L)^{1/n}$ then $\mathbf{v}$ is (a multiple of) the shortest vector in the lattice. A further consequence of [Ajt06] is that, for a "random" lattice of rank n, there exists a lattice basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$ of L such that $\|\mathbf{b}_i\| \approx \sqrt{n/(2\pi e)}\det(L)^{1/n}$ for all $1 \leq i \leq n$.

Let $1/4 < \delta < 1$. A basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$ for a lattice L is δ-LLL-reduced if the Gram-Schmidt vectors $\mathbf{b}_i^*$ satisfy $|\mu_{i,j}| \leq 1/2$ for $1 \leq j < i \leq n$ and

$$\|b_i^*\|^2 \geq \left(\delta - \mu_{i,i-1}^2\right)\|b_{i-1}^*\|^2$$

for $2 \leq i \leq n$, where $\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle$. It is known that an LLL-reduced lattice basis satisfies

$$\det(L) \leq \prod_{i=1}^{n} \|\mathbf{b}_i\| \leq 2^{n(n-1)/4}\det(L)$$

and $\|\mathbf{b}_1\| \leq 2^{(n-1)/2}\lambda_1(L)$, where $\lambda_1(L)$ is the length of the shortest non-zero vector of L. Furthermore, it is known that an LLL-reduced basis satisfies

$$\|\mathbf{b}_i\| \leq \left(2^{n(n-1)/4}\det(L)\right)^{1/(n+1-i)}$$

for $1 \leq i \leq n$.

It is folklore that LLL performs better on average than these worst-case bounds suggest. Nguyen and Stehlé [NgSt06] have studied the behaviour of LLL on "random" lattices and have hypothesised that an LLL-reduced basis satisfies the improved bound

$$\|\mathbf{b}_1\| \leq (1.02)^n \det(L)^{1/n}.$$

Based on this we suppose that $\|\mathbf{b}_1\| \leq (1.04)^n \lambda_1(L)$. Figure 4 of [NgSt06] shows that $\|\mathbf{b}_{i+1}^*\| \leq \|\mathbf{b}_i^*\|$ almost always, and certainly $\|\mathbf{b}_{i+1}^*\| \leq 1.2\|\mathbf{b}_i^*\|$ with overwhelming probability. Hence, we make the heuristic assumption that, for "random" lattices, $\|\mathbf{b}_i^*\| \leq \|\mathbf{b}_1^*\|$ for all $2 \leq i \leq n$. From this it is easy to show that, for $2 \leq i \leq n$,

$$\|\mathbf{b}_i\| \leq \sqrt{1 + (i-1)/4}\|\mathbf{b}_1\|.$$

In other words, on average LLL produces a basis that behaves close to the Gaussian heuristic. The analysis of lattice attacks in [DGHV10,CS15] is under an assumption of this type. We formalise this with the below heuristic assumption.

Assumption 1 *Let L be a "random" lattice of rank n and let $\mathbf{b}_1, \ldots, \mathbf{b}_n$ be an LLL-reduced basis for L. Then*

$$\|\mathbf{b}_i\| \leq \sqrt{i}(1.02)^n \det(L)^{1/n}.$$

for all $1 \leq i \leq n$.

3 Simultaneous Diophantine approximation approach (SDA)

In this and the following two sections we describe the three most successful lattice-based algorithms to solve the ACD problem when the error term is too large for exhaustive search and when sufficiently many samples are available.

The basic idea of this attack is to note that if $x_i = pq_i + r_i$ for $1 \leq i \leq t$, where r_i is small, then

$$\frac{x_i}{x_0} \approx \frac{q_i}{q_0}$$

for $1 \leq i \leq t$. In other words, the fractions q_i/q_0 are an instance of simultaneous Diophantine approximation to x_i/x_0. This was first noted by Howgrave-Graham (see Section 2 of [HG01]) and was further developed in Section 5.2 of [DGHV10]. Once q_i is known one can compute $r_i \equiv x_i \pmod{q_i}$ and hence $(x_i - r_i)/q_i = p$ and so the problem is solved. Note that this attack does not benefit significantly from having an exact sample $x_0 = pq_0$, so we do not assume that such a sample is given.

Following [DGHV10] we build a lattice L of rank $t+1$ generated by the rows of the basis matrix

$$\mathbf{B} = \begin{pmatrix} 2^{\rho+1} & x_1 & x_2 & \cdots & x_t \\ & -x_0 & & & \\ & & -x_0 & & \\ & & & \ddots & \\ & & & & -x_0 \end{pmatrix}. \tag{2}$$

Note that L contains the vector

$$\begin{aligned}\mathbf{v} &= (q_0, q_1, \cdots, q_t)\mathbf{B} \\ &= (2^{\rho+1}q_0, q_0x_1 - q_1x_0, \cdots, q_0x_t - q_tx_0) \\ &= (q_0 2^{\rho+1}, q_0r_1 - q_1r_0, \cdots, q_0r_t - q_tr_0)\end{aligned}$$

Since $q_0 \approx 2^{\gamma-\eta}$ the Euclidean norm of $\mathbf{v}$ is approximately $\sqrt{t+1}\,2^{\gamma-\eta+\rho+1}$. We give a more precise estimate in Lemma 1. We call this vector the **target vector**.

Since the basis matrix $\mathbf{B}$ of the lattice L is given in upper triangular form, the determinant of L is easily computed as $\det(L) = 2^{\rho+1}x_0^t$. Hence, if

$$\sqrt{t+1}\,2^{\gamma-\eta+\rho+1} < \sqrt{\frac{t+1}{2\pi e}}\det(L)^{1/(t+1)}$$

then we expect by the Gaussian heuristic that the target vector $\mathbf{v}$ is the shortest non-zero vector in the lattice. The attack is to run a lattice basis reduction algorithm to get a candidate $\mathbf{w}$ for the shortest non-zero vector. One then divides the first entry of $\mathbf{w}$ by $2^{\rho+1}$ to get a candidate solution value for q_0 and then computes the remaining values q_i. One then computes the r_i and checks if they are sufficiently small and that $(x_i - r_i)/q_i$ all give the same value for p, in which case the attack has succeeded. We call this the **SDA algorithm**.

This method is analysed in Section 5.2 of [DGHV10], where it is argued that if $t < \gamma/\rho$ then there are likely many vectors of around the same size or smaller as the desired vector. Hence it is required that $t > \gamma/\rho$ to have any chance for this method to succeed, even disregarding the difficulties of lattice reduction methods to find the shortest vector.

We make some specific remarks, that are relevant for comparing this attack with the other attacks. First, this attack only requires a single short vector, not a large number of short vectors. Second, the attack is heuristic because we are assuming that $\mathbf{v}$ is the only vector in the lattice that is shorter than the length predicted by the Gaussian heuristic. However, this seems to be a relatively mild heuristic in practice. Third, if we wish to use LLL to break the system we require $\mathbf{v}$ to be shorter by an exponential factor than the second successive minimum. In other words, we need $2^{t/2}\|\mathbf{v}\| \leq \sqrt{n}\det(L)^{1/(t+1)}$. The factor $2^{t/2}$ can be reduced using heuristics on the average-case behaviour of LLL, or by using more powerful basis reduction algorithms such as BKZ.

We now repeat the analysis of [DGHV10], with an eye to the Cheon-Stehlé-ACD parameters, and also using a more precise estimate of the target vector than was given in previous work.

Lemma 1. *The expected length of the target vector* $\mathbf{v}$ *is*

$$0.47\frac{\sqrt{t+1}}{p}2^{\rho+\gamma}.$$

Proof. Note that both the q_i and the r_i are random variables on $\mathbb{Z}$ with distributions

$$q_i \leftarrow \text{Uni}\{0, \ldots, \lfloor p^{-1}2^{\gamma}\rfloor\} \quad \text{and} \quad r_i \leftarrow \text{Uni}\{-2^{\rho}, \ldots 2^{\rho}\},$$

where Uni denotes the uniform distribution and $\leftarrow$ represents sampling from a distribution. It follows that $\mathbf{E}\left(q_i^2\right) \approx \frac{1}{3}p^{-2}2^{2\gamma}$, $\mathbf{E}(r_i) = 0$ and $\mathbf{E}\left(r_i^2\right) \approx \frac{1}{3}2^{2\rho}$. Furthermore, all of these random variables are independent, so we have

$$\begin{aligned}\mathbf{E}\left((q_0 r_i - q_i r_0)^2\right) &= \mathbf{E}\left(q_0^2 r_i^2\right) + \mathbf{E}\left(q_i^2 r_0^2\right) - 2\mathbf{E}\left(q_0 r_i q_i r_0\right) \\ &= \mathbf{E}\left(q_0^2\right)\mathbf{E}\left(r_i^2\right) + \mathbf{E}\left(q_i^2\right)\mathbf{E}\left(r_0^2\right) - 2\mathbf{E}\left(q_0 q_i\right)\mathbf{E}\left(r_i\right)\mathbf{E}\left(r_0\right) \\ &\approx \tfrac{2}{9}p^{-2}2^{2(\rho+\gamma)}.\end{aligned}$$

It follows that the root mean squared length of $\mathbf{v}$ is given by

$$\mathbf{E}\left(|\mathbf{v}|^2\right)^{\frac{1}{2}} \approx \left(\tfrac{2}{9}\right)^{\frac{1}{2}}(t+1)^{\frac{1}{2}}p^{-1}2^{(\rho+\gamma)} \approx 0.47\,(t+1)^{\frac{1}{2}}p^{-1}2^{(\rho+\gamma)}.$$

This completes the proof. □

The estimate for the length of $\mathbf{v}$ given in [DGHV10] is $(t+1)^{\frac{1}{2}}2^{(\rho+\gamma-\eta)}$, that is to say about twice the above approximation (taking $p \approx 2^{\eta}$).

The attacker hopes that the lattice is a "gap lattice" in the sense that the first minimum $\lambda_1(L) = \|\mathbf{v}\|$ is much shorter than the length $\lambda_2(L)$ of the next shortest vector in L independent of $\mathbf{v}$. We apply the Gaussian heuristic to estimate

$$\lambda_2(L) \approx \sqrt{(t+1)/(2\pi e)}\det(L)^{1/(t+1)} \approx \sqrt{(t+1)/(2\pi e)}2^{(\rho+1+\gamma t)/(t+1)}.$$

We know LLL succeeds in finding $\mathbf{v}$ if $\lambda_2(L) > 2^{t/2}\lambda_1(L)$, but on average one has a smaller exponential factor (or one can use BKZ or other algorithms to find short vectors). Hence, the target vector is the shortest vector in the lattice and is found by LLL if

$$0.47\sqrt{t+1}(1.04)^{t+1}2^{\gamma+\rho-\eta} < \sqrt{(t+1)/(2\pi e)}2^{(\rho+1+\gamma t)/(t+1)}. \tag{3}$$

Van Dijk et al [DGHV10] show that, in order that $\mathbf{v}$ is heuristically the shortest vector in the lattice, one needs to use $t > \gamma/\eta$ samples and a lattice of dimension $t+1$. Their analysis assumes that ρ is small and is not helpful when considering the Cheon-Stehlé variant of the problem. Hence, we re-consider their analysis. Ignoring constants in the above equation, we find that a necessary (not sufficient) condition for the algorithm to succeed is

$$t + 1 > \frac{\gamma - \rho}{\eta - \rho}. \tag{4}$$

For the Cheon-Stehlé variant we may have ρ close to η (e.g., $\rho = \lambda$ and $\eta = \lambda + 10\log(\lambda)$), which means the required dimension can grow very fast even with relatively small values for γ. More precisely, [CS15] suggests

$$(\rho, \eta, \gamma) = (\lambda, \lambda + d\log(\lambda), \Omega(d^2\lambda\log(\lambda)))$$

where d is the circuit depth and λ is the security parameter. Taking $\lambda = 80$ and $d = 10$ and setting $\Omega(x) = x$ we have $(\rho, \eta, \gamma) = (80, 143, 50575)$, which is very modest compared with the parameters in [DGHV10]. However, for these values, $(\gamma - \rho)/(\eta - \rho) \geq 800$, should be large enough to prevent any practical lattice attack. These arguments

therefore confirm the analysis from [CS15] that their approach should provide more efficient parameters for homomorphic encryption.

The above analysis ignored some terms, so as a final remark we justify why these approximations are reasonable. Equation (3) states that we need

$$(0.47)\sqrt{2\pi e}(1.04)^{t+1}2^{\rho-\eta} < 2^{(\rho+1-\gamma)/(t+1)}.$$

Taking logs and noting that $\rho < \eta < \gamma$ gives

$$\eta - \rho - 1 - (t+1)\log_2(1.04) > (\gamma - \rho - 1)/(t+1) > 0.$$

Writing $A = \log_2(1.04), B = \eta - \rho - 1$ and $C = \gamma - \rho - 1$ this is

$$A(t+1)^2 - B(t+1) + C < 0.$$

We are interested in the range of t for which this occurs, so it is natural to seek the smallest $x > 0$ for which $Ax^2 - Bx + C = 0$. Note that $A \approx 0.06$, $C \approx \gamma$ and $B^2 \approx \eta^2$. If we assume that $\eta > 4\rho$ and $\eta^2 > \gamma$ then $0 < 4AC/B^2 \ll 1$. Using $B^2 - 4AC = B^2(1 - 4AC/B^2)$ and $\sqrt{1-2\epsilon} \approx 1 - \epsilon$ for small ϵ we compute

$$\sqrt{B^2 - 4AC} \approx B(1 - 2AC/B^2).$$

The smallest choice for t that satisfies the inequality is therefore close to

$$\frac{B - \sqrt{B^2 - 4AC}}{2A} \approx \frac{B - B(1 - 2AC/B^2)}{2A} = C/B = \frac{\gamma - \rho - 1}{\eta - \rho - 1}.$$

One sees that this agrees with the original estimate, and so within that range of parameters, the term $(1.04)^{t+1}$ does not have any significant effect on the performance of the algorithm.

4 Orthogonal based approach (OL)

Nguyen and Stern (see for example [NgSt01]) have demonstrated the usefulness of the orthogonal lattice in cryptanalysis, and this has been used in several ways to attack the ACD problem. Appendix B.1 of [DGHV10] gives a method based on vectors orthogonal to $(x_1, \ldots, x_t)$. Their idea is that the lattice of integer vectors orthogonal to $(x_1, \ldots, x_t)$ contains the sublattice of integer vectors orthogonal to both $(q_1, \ldots, q_t)$ and $(r_1, \ldots, r_t)$. Later in Appendix B.1 of [DGHV10] a method is given based directly on vectors orthogonal to $(1, -r_1/R, \ldots, -r_t/R)$, where $R = 2^\rho$. Ding and Tao [DT14] have given a method based on vectors orthogonal to $(q_1, \ldots, q_t)$. Cheon and Stehlé [CS15] have considered the second method from Appendix B.1 of [DGHV10].

Our analysis (as with that in [DGHV10]) and experiments suggest these methods all essentially have the same performance in both theory and practice. Indeed, all three methods end up computing short vectors that are orthogonal to $(q_1, \ldots, q_t)$ and some vector related to the error terms r_i, for example see Lemma 3. Hence, in this paper we follow [DGHV10,CS15] and study the use of vectors orthogonal to $(1, -r_1/R, \ldots, -r_t/R)$.

These attacks do not benefit significantly from having an exact sample $x_0 = pq_0$ so we do not use it.

Let $R = 2^\rho$ be an upper bound on the absolute value of the errors r_i in $x_i = pq_i + r_i$. Let L be a lattice in $\mathbb{Z}^{t+1}$ with basis matrix

$$\mathbf{B} = \begin{pmatrix} x_1 & R & & & \\ x_2 & & R & & \\ x_3 & & & R & \\ \vdots & & & & \ddots \\ x_t & & & & & R \end{pmatrix}. \tag{5}$$

Clearly the rank of B is t. The lattice volume was estimated in previous works, but we give an exact formula.

Lemma 2. *The Gram matrix $\mathbf{B}\mathbf{B}^T$ of L is of the form $R^2\mathbf{I}_t + \mathbf{x}^T\mathbf{x}$ where $\mathbf{x} = (x_1, \ldots, x_t)$ and $\mathbf{I}_t$ is the $t \times t$ identity matrix. The volume of the lattice is $R^{t-1}\sqrt{R^2 + x_1^2 + \cdots + x_t^2}$.*

Proof. The claim about $\mathbf{B}\mathbf{B}^T$ is easily checked by induction. Writing this as $\mathbf{B}\mathbf{B}^T = \mathbf{A} + \mathbf{x}^T\mathbf{x}$ where $\mathbf{A} = R^2\mathbf{I}_t$ is invertible, the matrix determinant lemma states that $\det(\mathbf{B}\mathbf{B}^T) = \det(\mathbf{A})(1 + \mathbf{x}\mathbf{A}^{-1}\mathbf{x}^T)$. Since $\det(\mathbf{A}) = R^{2t}$ and $\mathbf{A}^{-1} = \frac{1}{R^2}\mathbf{I}_t$ we find

$$\det(\mathbf{B}\mathbf{B}^T) = R^{2t}\left(1 + \frac{x_1^2 + \cdots + x_t^2}{R^2}\right) = R^{2(t-1)}(R^2 + x_1^2 + \cdots + x_t^2).$$

The final claim comes from the fact that the lattice volume is $\sqrt{\det(\mathbf{B}\mathbf{B}^T)}$. □

Any vector $\mathbf{v} = (v_0, v_1, \cdots, v_t) \in L$ is of the form

$$\mathbf{v} = (u_1, \cdots, u_t)\mathbf{B} = \left(\sum_{i=1}^{t} u_i x_i, u_1 R, u_2 R, \cdots, u_t R\right),$$

where $u_i \in \mathbb{Z}$. The main observation of Van Dijk et al. [DGHV10] is

$$v_0 - \sum_{i=1}^{t} \frac{v_i}{R} r_i = \sum_{i=1}^{t} u_i x_i - \sum_{i=1}^{t} \frac{u_i R}{R} r_i = \sum_{i=1}^{t} u_i (x_i - r_i) = 0 \pmod{p}. \tag{6}$$

Since we don't know p, we wish to have a linear equation over $\mathbb{Z}$. The equation holds if $|v_0 - \sum_{i=1}^{t} \frac{v_i}{R} r_i| < p/2$. The following lemma gives a bound on $\mathbf{v}$ that implies we get an integer equation as desired.

Lemma 3. *Let $\mathbf{v} = (u_0, u_1, u_2, \cdots, u_t)\mathbf{B}$. Let $||\mathbf{v}|| \leq 2^{\eta - 2 - \log_2(t+1)}$. Then*

$$|v_0 - \sum_{i=1}^{t} u_i r_i| < p/2 \quad \textit{and} \quad \sum_{i=1}^{t} u_i q_i = 0.$$

Proof. Let $\mathbf{v} = (v_0, v_1, \cdots, v_t) = (\sum_{i=1}^{t} u_i x_i, u_1 R, u_2 R, \cdots, u_t R)$ and let $N = ||\mathbf{v}||$. Then $|v_0| \leq N$ and $|u_i r_i| \leq |u_i R| \leq N$ for $1 \leq i \leq t$. Thus

$$\left| v_0 - \sum_{i=1}^{t} u_i r_i \right| \leq |v_0| + \sum_{i=1}^{t} |u_i r_i| \leq (t+1)N.$$

Since $N \leq 2^{\eta - 2 - \log_2(t+1)}$, we have $(t+1)N < 2^{\eta-2} < p/2$ since $p > 2^{\eta-1}$. Hence $|v_0 - \sum_{i=1}^{t} u_i r_i| < p/2$.

To prove $\sum_{i=1}^{t} u_i q_i = 0$, suppose $\sum_{i=1}^{t} u_i q_i \neq 0$ so that $p|\sum_{i=1}^{t} u_i q_i| \geq p > 2^{\eta-1}$. Since $x_i = pq_i + r_i$, we have

$$\begin{aligned} p|\sum_{i=1}^{t} u_i q_i| &= |\sum_{i=1}^{t} u_i (x_i - r_i)| \\ &\leq |\sum_{i=1}^{t} u_i x_i| + |\sum_{i=1}^{t} u_i r_i|. \end{aligned}$$

But, by the previous argument, this is $\leq (t+1)N < 2^{\eta-1}$, which is a contradiction. □

In other words, every short enough vector $\mathbf{v}$ in the lattice gives rise to an inhomogeneous equation $v_0 = \sum u_i r_i$ in the t variables r_i, and a homogeneous equation $\sum_i u_i q_i = 0$ in the t variables q_i. There are therefore two approaches to solve the system. The papers [DGHV10,CS15,DT14] use t inhomogeneous equations and solve for the r_i, but we believe it is simpler and faster to use $t-1$ equations and then find the kernel of the matrix formed by them to solve for $(q_1, \ldots, q_t)$. We call these methods the **OL algorithm**.

Hence, it suffices to have $t-1$ linearly-independent vectors in the lattice L that satisfy the bound of Lemma 3. Hence we run lattice reduction and take the $t-1$ smallest vectors in the output basis (they are linearly independent as required).

To analyse the method we use Assumption 1. This shows that one can compute using LLL $t-1$ linearly-independent vectors of the correct size as long as

$$\sqrt{t}(1.02)^t \det(L)^{1/t} \leq 2^{\eta - 2 - \log_2(t+1)}.$$

By Lemma 2 we approximate $\det(L)$ by $2^{\rho(t-1)+\gamma}$. Hence, the condition for success is

$$4\sqrt{t(t+1)}(1.02)^t 2^{\rho + (\gamma-\rho)/t} \leq 2^{\eta}.$$

Following the analysis in [DGHV10,CS15], if one ignores constants and exponential approximation factors in the lattice reduction algorithm, then a necessary condition on the dimension is $t \geq (\gamma - \rho)/(\eta - \rho)$, which is the same as equation (4) for the SDA

method.[3] Hence, we deduce that the OL method is not more powerful than the SDA method. Our experimental results confirm this, though they suggest the OL method is slightly faster (due to the smaller size of entries in the basis matrix defining the lattice).

We give one remark about the CRT-ACD problem. Recall that each ACD instance x_i in this problem satisfies $x_i \equiv r_{i,j} \pmod{p_j}$ for many primes p_i, where $r_{i,j}$ is small. Hence there are many variants of equation (6)

$$v_0 - \sum_{i=1}^{t} \frac{v_i}{R} r_{i,j} = \sum_{i=1}^{t} u_i x_i - \sum_{i=1}^{t} \frac{u_i R}{R} r_{i,j} = \sum_{i=1}^{t} u_i (x_i - r_{i,j}) = 0 \pmod{p}_j.$$

In practice this causes the lattice method to be much less effective, since different short vectors may correspond to different choices of prime p_j and hence different values for the $r_{i,j}$. It remains an open problem to analyse the security of this variant of the ACD problem.

5 Multivariate polynomial approach (MP)

Howgrave-Graham [HG01] was the first to consider reducing the approximate common divisor problem to the problem of finding small roots of multivariate polynomial equations. The idea was further extended in Appendix B.2 of van Dijk et al [DGHV10]. Finally, a detailed analysis was given by Cohn and Heninger [CH13]. However, the paper [CH13] focusses on the case when a small number of ACD samples are available, it uses worst-case rather than average-case LLL bounds, and it contains no comparison against the orthogonal lattice approaches. Hence, in this section we give a revised analysis of the algorithm and report on experiments with it. Our heuristic analysis and experimental results suggest that the best choice of parameters for the multivariate approach is to use linear polynomials, in which case the algorithm is equivalent to the orthogonal lattice method. In other words, we find that the multivariate approach seems to have no advantage over the orthogonal lattice method when attacking ACD instances coming from crypto applications.

The multivariate approach can be applied to both the full and partial ACD problems, but it is simpler to explain and analyse for the partial ACD problem. Hence, in this section we restrict to this case only.[4]

We change notation from the rest of the paper to follow more closely the notation used in [CH13]. Note that the symbols X_i are variables, not ACD samples. Hence, let $N = pq_0$ and let $a_i = pq_i + r_i$ for $1 \leq i \leq m$ be our ACD samples, where $|r_i| \leq R$ for some given bound R. The idea is to construct a polynomial $Q(X_1, X_2, \ldots, X_m)$ in m variables such that $Q(r_1, \cdots, r_m) \equiv 0 \pmod{p^k}$ for some k. The parameters m and k are optimised later. In [CH13], such a multivariate polynomial is constructed as integer

[3] There is no need to repeat the more careful analysis we already did for SDA, since we are lower-bounding the OL method by the SDA method.

[4] Since the orthogonal lattice method performs equally well for both full and partial ACD, it suffices to compare the methods in the case most favourable to the multivariate polynomial approach.

linear combinations of the products

$$(X_1 - a_1)^{i_1} \cdots (X_m - a_m)^{i_m} N^{\ell}$$

where ℓ is chosen such that $i_1 + \cdots + i_m + \ell \geq k$.

An additional generality is to choose a degree bound $t \geq k$ (do not confuse this with the use of the symbol t previously) and impose the condition $i_1 + \cdots + i_m \leq t$. The value t will be optimised later.

The lattice L is then defined by the coefficient row vectors of the polynomials

$$f_{[i_1,\ldots,i_m]}(X_1, \ldots, X_m) = (RX_1 - a_1)^{i_1} \cdots (RX_m - a_m)^{i_m} N^{\ell}, \tag{7}$$

such that $i_1 + \cdots + i_m \leq t$ and $\ell = \max(k - \sum_j i_j, 0)$. For example, the values $(t, m, k) = (3, 2, 1)$ lead to the following basis matrix.

$$\mathbf{B} = \begin{array}{c} \\ f_{[0,0]} \\ f_{[1,0]} \\ f_{[0,1]} \\ f_{[2,0]} \\ f_{[1,1]} \\ f_{[0,2]} \\ \vdots \\ f_{[0,3]} \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & X_1 & X_2 & X_1^2 & X_1X_2 & X_2^2 & \ldots & X_2^3 \end{array} \\ \begin{pmatrix} N & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\ -a_1 & R & 0 & 0 & 0 & 0 & \ldots & 0 \\ -a_2 & 0 & R & 0 & 0 & 0 & \ldots & 0 \\ a_1^2 & -2a_1R & 0 & R^2 & 0 & 0 & \ldots & 0 \\ a_1a_2 & -a_2R & -a_1R & 0 & RR & 0 & \ldots & 0 \\ a_2^2 & 0 & -2a_2R & 0 & 0 & R^2 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_2^3 & 0 & 3a_2^2R & 0 & 0 & -3a_2R^2 & \ldots & R^3 \end{pmatrix} \end{array}. \tag{8}$$

It is shown in [CH13] that L has dimension $d = \binom{t+m}{m}$ and determinant

$$\det(L) = R^{\binom{t+m}{m}\frac{mt}{m+1}} N^{\binom{k+m}{m}\frac{k}{m+1}} = 2^{d\frac{\rho mt}{m+1} + \binom{k+m}{m}\frac{\gamma k}{m+1}}.$$

A natural choice for R is 2^{ρ}.

Let $\mathbf{v}$ be a vector in L. One can interpret $\mathbf{v} = (v_{j_1,\cdots,j_m} R^{j_1+\cdots+j_m})$ as the coefficient vector of a polynomial

$$Q(X_1, \ldots, X_m) = \sum_{j_1,\cdots,j_m} v_{j_1,\cdots,j_m} X_1^{j_1} \cdots X_m^{j_m}.$$

If $|Q(r_1, \cdots, r_m)| < p^k$ then we have $Q(r_1, \cdots, r_m) = 0$ over the integers, so we need to bound $|Q(r_1, \cdots, r_m)|$. Note that

$$\begin{aligned} |Q(r_1, \cdots, r_m)| &\leq \sum_{j_1,\cdots,j_m} |v_{j_1\cdots j_m}||r_1|^{j_1} \cdots |r_m|^{j_m} \\ &\leq \sum_{j_1,\cdots,j_m} |v_{j_1\cdots j_m}| R^{j_1} \cdots R^{j_m} \\ &= \|\mathbf{v}\|_1. \end{aligned}$$

Hence, if $\|\mathbf{v}\|_1 < p^k$ then we have a suitable polynomial. We call a vector $\mathbf{v} \in L$ such that $\|\mathbf{v}\|_1 < p^k$ a **target vector**. We will need (at least) m algebraically independent target vectors to be able to perform elimination (using resultants or Gröbner basis method)

to reduce to a univariate polynomial equation and hence solve for $(r_1, \ldots, r_m)$. One then computes $p = \gcd(N, a_1 - r_1)$. Note that solving multivariate polynomial equations of degree greater than one in many variables is very time consuming and requires significant memory. In practice, the elimination process using Gröbner basis methods is faster if the system is overdetermined, so we generally use more than m polynomials. We call this process the **MP algorithm**.

We remark that the case $(t, k) = (1, 1)$ gives essentially the same lattice as in equation (5) and so this case of the MP algorithm is the same as the orthogonal lattice attack (this was already noted in [DGHV10] and is also mentioned in Section 6 of [CH13]). Because of this, one can always say that the MP attack is at least as good as the orthogonal lattice attack. But the interesting question is whether any other choices of parameters for the MP algorithm give rise to better attacks.

5.1 The Cohn-Heninger Analysis

Cohn and Heninger [CH13] give a heuristic theoretical analysis of the MP algorithm and suggest optimal parameter choices (t, m, k). Their paper does this very briefly and omits some details, so we sketch their approach here.

Cohn and Heninger [CH13] introduce a parameter $\beta = \eta/\gamma \ll 1$ so that $p \geq N^\beta$. They work with the equation

$$\frac{mt \log_2(R)}{(m+1)k} \; + \; \frac{\gamma k^m}{(m+1)t^m} \; < \beta\gamma = \eta \tag{9}$$

which is a version of equation (10) below, with some terms deleted. They make a number of simplifying assumptions, assume that the best results will come from taking t large, and impose the asymptotic relationship $t \approx \beta^{-1/m}k$, which means that $t \gg k$. Their method allows errors up to $R = \gamma\beta^{(m+1)/m}$. They require $\beta^2 \log(N) \gg 1$ for the method to work[5], which is equivalent to $\eta^2 \gg \gamma$. The lattice dimension in their method is $\binom{t+m}{m} = O(t^m) = O(\beta^{-1}k^m) = O(\gamma/\eta)$, and so yet again we encounter the same dimension bound as the previous methods (at least, when ρ is small). The main "heuristic theorem" of [CH13] can be stated as: for fixed m, if $\beta = \eta/\gamma$ where $\eta^2 \gg \gamma$ and $\rho = \log_2(R) < \eta(1 + o(1))\beta^{1/m}$ then one can solve the ACD problem in polynomial time. The claim of polynomial-time complexity is correct, but does not imply that the MP approach is better than the SDA or OL approaches: The input size is proportional to γ and all the algorithms use lattices of dimension approximately γ/η when ρ is small, so they are all polynomial time if they return a correct solution to the problem.

The condition $\eta^2 \gg \gamma$ already means the MP attack can be avoided in practice relatively easily. We remark that the orthogonal lattice method does not have any such hard limit on its theoretical feasibility. However, in practice the restriction $\eta^2 \gg \gamma$ is not so different from the usual condition that the dimension must be at least γ/η: if $\gamma > \eta^2$ then the required dimension would be at least η, which is infeasible for lattice reduction algorithms for the sort of parameters used in practice.

[5] It is mentioned in Section 2.1 of [CH13] that this can be relaxed to $\beta^{1+\epsilon} \log(N) \gg 1$ if a lattice reduction algorithm with a sub-exponential approximation factor is available.

It is also important to consider the parameters of interest in the Cheon-Stehlé scheme. Hence we now suppose $\rho \approx \eta$ (e.g., $\rho/\eta = 0.9$) and $\gamma = \eta^{1+\delta}$ for some $\delta > 0$ and ask if the MP method can be better than the OL method in this setting. The condition $t\rho < k\eta$ implies that $t \approx k$, (recall that $t \geq k$) in which case $\binom{k+m}{m} \approx d = \binom{t+m}{m}$ and so the bound from equation (12) suggests the MP approach has no advantage over other methods for parameters of this type. Our experimental results confirm this (see Table 1).

5.2 Improved Analysis

We closely follow the analysis in [CH13], but we use average-case bounds on LLL (Assumption 1) rather than the worst-case bounds of equation (2.2).[6] Another of our main contributions is to consider the parameters more generally, unlike in [CH13] where it was assumed that the optimal solution would be to take $t, k > 1$ (e.g., they write "If t and k are large, then..." and "we take $t \gg m$ and ... $k^m \approx \beta t^m$"). Instead, we will argue that the best choices for the MP algorithm are $(t, k) = (1, 1)$. In other words, the MP method seems to have no advantage over the orthogonal lattice method.

The MP algorithm succeeds if we can produce m vectors in the lattice such that $\|\mathbf{v}\|_1 < p^k$. Note that the heuristics immediately differ between the cases $t = 1$ and $t > 1$. When $t > 1$ the number of target vectors required is much smaller than the dimension $d = \dim(L) = \binom{t+m}{m}$, however we require the corresponding polynomials to be algebraically independent which is a much stronger assumption than linear independence of the corresponding vectors. On the other hand, when $t = 1$ we require $m = d - 1$ short vectors so need a stronger assumption on the shape of the lattice basis, however it suffices to have linearly independent vectors to complete the attack.

Using $\|\mathbf{v}\|_1 \leq \sqrt{d}\|\mathbf{v}\|$ (where the latter is the Euclidean norm) and the bounds from Assumption 1 we have that an LLL-reduced basis satisfies

$$\|\mathbf{b}_i\|_1 \leq d(1.02)^d \det(L)^{1/d}$$

where d is the dimension of the lattice. If this bound is less than $p^k \approx 2^{\eta k}$ then we will have enough target vectors. Hence we need

$$d^d(1.02)^{d^2} \det(L) < 2^{\eta k d}$$

and so we need

$$d\log_2(d) + d^2\log_2(1.02) + d\rho\frac{mt}{m+1} + \gamma\binom{k+m}{m}\frac{k}{m+1} < k\eta d. \tag{10}$$

We now derive some useful necessary conditions for the algorithm to succeed. Noting that $\frac{mt}{m+1} \approx t$ we see that it is necessary to have

$$t\rho < k\eta, \tag{11}$$

[6] This does not have a major effect, since the analysis in [CH13] ignored several "nuisance factors" which boil down to the same thing as our assumption.

and so t cannot grow too fast compared with k. Similarly, we see it is necessary that $\gamma\binom{k+m}{m}\frac{k}{m+1} < k\eta d$ which is equivalent to

$$d = \binom{t+m}{m} > \frac{\gamma}{\eta}\binom{k+m}{m}\frac{1}{m+1}. \tag{12}$$

When $k = 1$ then the right hand side is equal to γ/η, but it gets steadily larger as k grows. Since γ/η is large, this shows that t has to be significantly larger than k, or else m has to be very large. At the very least, this condition demonstrates that the MP method does not overcome the minimal degree bound γ/η we already saw for the SDA and OL methods. (In the case $(t, k) = (1, 1)$ equation (12) simply becomes $m+1 > \gamma/\eta$ which we have already seen in Sections 4 and 3.)

We now attempt to give some further justification of our claim that the Cohn and Heninger analysis with $t > 1$ does not give any advantage over the case $t = 1$. To do this, suppose $\gamma = \eta^{1+\delta}$ for some $0 < \delta < 1$ so that $\beta = 1/\eta^{\delta}$ and $\beta^2\gamma = \eta^{1-\delta} \gg 1$ as required. The analysis in [CH13] suggests that m and t should be large, but does not seem to require k to grow large, hence we take $k = 1$. Equation (9) then becomes, recalling that $k^m = \beta t^m$,

$$\frac{mt\log_2(R)}{(m+1)} + \frac{\gamma\beta}{(m+1)} < \eta$$

It therefore suffices that the lattice dimension should be taken to be $d \approx t^m/m! = 1/(\beta m!) = \eta^{\delta}/m!$, giving $t \approx \eta^{\delta/m}$. Note that this lattice dimension is comparable with the lower bound $\gamma/\eta = \eta^{\delta}$ that we saw for the SDA and OL methods. From this we find that we can handle errors of bitlength $\log_2(R) \approx \eta/t$. There are many ways to choose parameters, but a reasonable choice seems to be to take $t \approx m$. This means that

$$d = \binom{t+m}{m} \approx \binom{2m}{m} \approx 4^m$$

and so $m \approx \log_4(d) \approx \delta\log_4(\eta)$. Finally, we have $\rho = \log_2(R) \approx \eta/(\delta\log_4(\eta))$.

To make this concrete, let us take $\eta = 1000$ and $\delta = 1/2$ so $\gamma = \eta^{1+\delta} = 31622$. Choosing $(t, m, k) = (5, 4, 1)$ gives a lattice of dimension $\binom{9}{4} = 126$ that can handle errors of size up to $R = 2^{1000/5} = 2^{200}$. For these values, we have $(\gamma - \rho)/(\eta - \rho) \approx 39$ which suggests the instance can be more easily solved using the SDA and OL methods or, equivalently, the MP method with parameters $(t, m, k) = (1, 50, 1)$. Our experimental results confirm these findings.

There are two further major advantages of the SDA and OL methods compared with the MP approach with $t > 1$. The first is that one can choose any value desired for the dimension, whereas in the MP method the dimension must be of the form $\binom{t+m}{m}$ and so it only takes certain values. The second is that the MP method with $t > 1$ requires solving systems of multivariate polynomial equations, and the cost of this stage can dwarf the cost of the lattice stage.

Table 1 gives a comparison of different parameters for the MP method. The left hand table is for $\eta = 100$ and varying values of γ. For different choices of (t, k) we determine the maximal ρ such that the MP algorithm with parameters (t, k) can solve the problem

Table 1. Comparison between different parameter choices (t, k) in the multivariate polynomial algorithm. The left hand table reports, for $\eta = 100$, the largest value for ρ that can be solved with reasonable probability for the given choice (γ, η, t, k, m). The right hand table compares running times for larger examples. $\dim(L)$, TLLL, and TGRB refer to the lattice dimension, running time (seconds) of the LLL algorithm and running of the Gröbner basis algorithms to solve the resulting polynomial systems respectively. The notation '**' indicates that the computation was aborted before a result was found after the fixed time period of a few minutes.

γ	$\rho_{\max}$	t	k	m	$\dim(L)$	TLLL	TGRB
150	95	1	1	30	31	0.020	0.020
	90	3	2	8	165	0.350	0.070
	85	4	3	4	70	0.220	0.040
300	90	1	1	30	31	0.030	0.130
	60	3	2	5	56	0.310	0.770
	60	4	3	4	70	4.150	15.150
600	80	1	1	30	31	0.070	0.020
	35	3	2	4	35	1.020	0.170
	10	4	3	3	35	2.930	4.640

γ	ρ	t	k	m	$\dim(L)$	TLLL	TGRB
300	10	1	1	4	5	0.020	0.000
		3	2	4	35	0.300	0.050
	50	1	1	6	7	0.010	0.010
		3	2	4	35	0.110	0.030
600	10	1	1	7	8	0.020	0.000
		3	2	4	35	1.070	6.100
	30	1	1	9	10	0.030	0.010
		3	2	4	35	1.020	5.330
1200	10	1	1	14	15	0.030	0.010
		3	2	5	56	14.130	347.200
	20	1	1	15	16	0.030	0.010
		3	2	5	56	13.890	297.820
2400	10	1	1	27	28	0.190	0.010
		3	2	5	56	32.710	**
	20	1	1	30	31	0.260	0.020
		3	2	5	56	32.480	**
5000	15	1	1	119	120	102.660	0.675
		2	1	10	66	10.380	**
	30	1	1	72	120	84.070	0.680
		2	1	11	78	18.010	**
8000	10	1	1	119	120	136.530	0.670
		2	1	14	120	219.140	**
		3	1	6	84	74.490	**
	15	1	1	119	120	145.770	0.670
		2	1	14	120	226.370	**
	20	1	1	1	120	164.750	0.670
		2	1	14	120	300.100	**

with high probability. This table shows that $(t, k) = (1, 1)$ allows to solve a wider range of parameters than other choices, which confirms our argument that $(t, k) = (1, 1)$ is better than other parameter choices. The second table considers larger values for γ and the aim of this table is to emphasise the considerable increase in the running time when using $t > 1$.

6 Experimental Observation

We have conducted extensive experiments with the SDA, OL and MP methods. For a small summary see Table 2. As with all lattice attacks, the running time depends mostly on the dimension of the lattice, and then on the size of the integers in the basis for the lattice. In general our experiments confirm that the OL method is the fastest and most effective algorithm for solving the ACD problem. For many more tables of experimental results we refer to Chapter 5 of [Geb16].

The parameters (ρ, η, γ) in Table 2 are selected according to the formula $(\lambda, \lambda + d\log(\lambda), d^2\lambda\log(\lambda))$ from [CS15], where λ is a security parameter and $d > 0$ is the depth of a circuit to allow decryption of depth d. We took $\lambda = 80$ and vary d from 1 to

Table 2. Comparison of orthogonal lattice (OL) and simultaneous Diophantine approximation (SDA) algorithms (note that the MP method with $(t, k) = (1, 1)$ is the same as the OL method).

η	γ	ρ	$\dim(L)$	OL time (seconds)	SDA time (seconds)
86	480	75	120	1.700	2.380
		70	40	0.110	0.200
		50	24	0.030	0.050
92	1920	50	56	1.540	5.020
98	4320	50	200	1242.640	4375.120
104	7680	50	200	3047.500	14856.630
110	12000	20	200	5061.760	27578.560
		10	200	3673.160	23428.410

5. Of course, we did not expect to solve this system quickly for the choice $\rho = \lambda$ (and our experiments confirmed this). We only report timings for slightly smaller values for ρ.

7 Pre-processing of the ACD samples

The most important factor in the difficulty of the ACD problem is the ratio γ/η, which is the size of the integers x_i relative to the size of p. If one can lower γ for the same p and without changing the size of the errors then one gets an easier instance of the ACD problem.

Hence, it is natural to consider a pre-processing step where a large number of initial samples $x_i = pq_i + r_i$ are used to form new samples $x'_j = pq'_j + r'_j$ with q'_j significantly smaller than q_i. The main idea we consider for doing this is by taking differences $x_k - x_i$ for $x_k > x_i$ and $x_k \approx x_i$. The essential property is that if $x_k \approx x_i$ then $q_k \approx q_i$ but r_k and r_i are not necessarily related at all. Hence $x_k - x_i = p(q_k - q_i) + (r_k - r_i)$ is an ACD sample for the same unknown p but with a smaller value for q and a similar sized error r. It is natural to hope that one can iterate this process until the samples are of a size suitable to be attacked by the orthogonal lattice algorithm.

This idea is reminiscent of the Blum-Kalai-Wasserman (BKW) algorithm [BKW03] for learning parity with noise (LPN). In that case we have samples $(\mathbf{a}, b)$ where $\mathbf{a} \in \mathbb{Z}_2^n$ is a vector of length n and $b = \mathbf{a} \cdot \mathbf{s} + e$, where $\mathbf{s} \in \mathbb{Z}_2^n$ is a secret and e is a noise term which is usually zero. we wish to obtain samples such that $\mathbf{a} = (1, 0, 0, \ldots, 0)$, or similar, and we do this iteratively by adding samples $(\mathbf{a}_k, b_k) + (\mathbf{a}_i, b_i)$ where some coordinates of $\mathbf{a}_k$ and $\mathbf{a}_i$ agree. The result is an algorithm with subexponential complexity $2^{n/\log(n)}$, compared with the naive algorithm (guessing all $\mathbf{s} \in \mathbb{Z}_2^n$) which has complexity 2^n. In our context we do not have $(q_i, pq_i + r_i)$ but only $x_i = pq_i + r_i$, however we can use the high-order bits of x_i as a proxy for the high order bits of q_i and hence perform a similar algorithm. A natural question is whether this leads to a faster algorithm for the ACD problem.

There are several approaches one might attempt. Let $x_1, \ldots, x_\tau$ be the initial list of γ-bit ACD samples.

1. (Preserving the sample size) Fix a small bound B (e.g., $B = 16$) and select B samples (without loss of generality call them $x_1, \ldots, x_B$) such that the leading

coefficients in base B are all distinct. For each of the remaining $\tau - B$ samples, generate a new sample by subtracting the one with the same leading coefficient. The result is $\tau - B$ samples each of size $\gamma - \log_2(B)$ bits.

2. (Aggressive shortening) Sort the samples $x_1 \leq x_2 \leq \cdots \leq x_\tau$ and, for some small threshold $T = 2^{\gamma-\mu}$, generate new samples by subtracting $x_{i+1} - x_i$ when this difference is less than T. The new samples are of size at most $\gamma - \mu$ bits, but there are far fewer of them.

7.1 Preserving the sample size

This first method is analysed briefly in [Geb16] and we give further informal discussion here. Suppose $B = 2^b$. After I iterations of the method we have generated approximately $\tau - IB$ samples, each of $\gamma - Ib$ bits. However, we must consider the size of the errors. The original samples $x_i = pq_i + r_i$ have errors $|r_i| \leq 2^\rho$, and the samples at iteration k are of the form

$$x = \sum_{i=1}^{2^k} c_i x_i \quad \text{where} \quad c_i = \pm 1$$

and so the error terms behave like a "random" sum of 2^k ρ-bit integers. Since the r_i are uniformly distributed in $[-2^\rho, 2^\rho]$, for large k the value $r = \sum_i c_i r_i$ has mean 0 and variance $\frac{1}{3}2^{2\rho+k}$. So we expect $|r| \leq 2^{\rho+k/2}$. Once $\rho + k/2 > \eta$ then the errors have grown so large that we have essentially lost all information about p, and the method is no good. Hence, an absolute upper limit on the number of iterations is $2(\eta - \rho)$. This means that after the final iteration the samples are reduced to bitlength no fewer than $\gamma - 2b(\eta - \rho)$ bits.

In terms of lattice attacks, an attack on the original problem requires a lattice of dimension roughly γ/η (assuming $\rho \ll \eta$). After the pre-processing we would need a lattice of dimension

$$\frac{\gamma - 2b(\eta - \rho)}{\eta} \approx \frac{\gamma}{\eta} - 2b.$$

Since a typical value for b is 8 or 16, we see that very little difference has been made to the problem.

7.2 Sample amplification

First experiments may lead one to believe that the aggressive shortening approach is fruitless. It is natural to choose parameters so that the lists are reduced at each iteration by some constant factor, and so the number of samples decreases exponentially in terms of the number of iterations. Eventually one has too few samples to run any of the previously mentioned lattice algorithms.

However, it turns out that a very simple strategy can be used in practice to increase the number of samples again. The idea is to generate new samples (that are still about the same bitlength) by taking sums/differences of the initial list of samples. This is similar to ideas used to amplify the number of samples for solving LPN or LWE [Lyu05].

Let $\mathcal{L} = \{x_1, \ldots, x_\tau\}$ be a list of ACD samples, with $x_k = pq_k + r_k$ having mean and variance given by $\mu = \mathbf{E}(x_k) = p\mathbf{E}(q_k) = 2^{\gamma-1}$ and variance given by

$$\begin{aligned}\text{Var}(x_k) &= p^2\text{Var}(q_k) + \text{Var}(r_k) = \tfrac{1}{3}2^{2(\gamma-1)} + \tfrac{1}{12}2^{2\rho} \\ &= \tfrac{1}{3}2^{2(\gamma-1)}\left(1 + 2^{-2(\gamma-\rho)}\right).\end{aligned}$$

We generate m random sums $S_1, \ldots, S_m$ of l elements of $\mathcal{L}$, that is to say we consider values of the form

$$S_k = \sum_{i=1}^{l} x_{k_i} \qquad [k = 1, \ldots, m],$$

which have mean and variance given by

$$\mathbf{E}(S_k) = l2^{\gamma-1} \text{ and } \text{Var}(S_k) = \tfrac{1}{3}l2^{2(\gamma-1)}\left(1 + 2^{-2(\gamma-\rho)}\right).$$

We note that (provided l is not too large) two such random variables S_k and $S_{k'}$ are usually sums of different ACD samples and so are usually independent. In any case, we can obtain many samples (with m potentially up to $\binom{\tau}{l}$) of a more peaked distribution, albeit with a slightly larger variance. Hence, not only have we created a much larger pool of samples, the non-uniform distribution of these samples makes them even more attractive for an algorithm based on computing neighbouring differences.

Recall that the next stage of the algorithm will be to sort the new samples $S_1, \ldots, S_m$ to obtain the list $S_{(1)} \leq \cdots \leq S_{(m)}$. We call these the *order statistics*. We then consider the neighbouring differences or *spacings* $T_k = S_{(k+1)} - S_{(k)}$ for $k = 1, \ldots, m-1$. In order to analyse the effectiveness of this approach we need to derive the statistical distribution of the spacings.

The statistical distribution of spacings arising from a general distribution is considered by Pyke [Pyk65], where it is shown that such generic spacings have Exponential distributions, and such an approach gives Lemma 4. We recall that the distribution function F for a random variable W on $\mathbb{R}$ is the monotonic function $F(w) = \mathbf{P}(W \leq w)$, which gives the density function $f = F'$ of W as the derivative of F (where this exists) and the inverse distribution function F^{-1} of W as the inverse function to F. Furthermore, a positive random variable $W \sim \text{Exp}(\lambda)$ is an Exponential random variable with (rate) parameter λ if its density function $f_W(w) = \lambda\exp(-\lambda w)$ $(w > 0)$, when $\mathbf{E}(W) = \lambda^{-1}$ and $\text{Var}(W) = \lambda^{-2}$, so an Exponential random variable has the same mean and standard deviation.

Lemma 4. *Suppose $Z_1, \ldots, Z_m$ are independent and identically distributed random variables on $\mathbb{R}$ with common distribution function F, inverse distribution function F^{-1} and density function $f = F'$. If $Z_{(1)} \leq \ldots \leq Z_{(m)}$ denote the order statistics of $Z_1, \ldots, Z_m$, then the k^{th} spacing $Z_{(k+1)} - Z_{(k)}$ is well-approximated for large m as an Exponential random variable with (rate) parameter $m\ f\left(F^{-1}\left(\frac{k}{m}\right)\right)$.*

Proof. Equations (4.9) and (4.10) of Pyke [Pyk65] show that the k^{th} spacing

$$Z_{(k+1)} - Z_{(k)} = \frac{1}{(m-k)}\frac{(1 - A_{k+1})}{f\left(F^{-1}\left(A_{k+1}\right)\right)}\ Y_k,$$

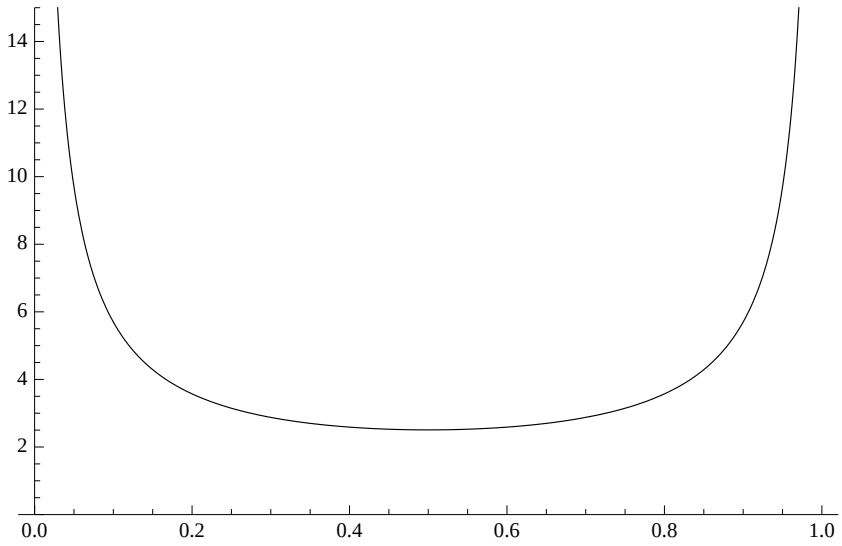

Fig. 1. Graph of the Function $H(u) = g\left(G^{-1}(u)\right)^{-1}$.

where $Y_k \sim \text{Exp}(1)$ is an Exponential random variable and A_{k+1} essentially lies between the k^{th} and $(k+1)^{\text{th}}$ order statistics of m random variables uniformly distributed on $(0,1)$. Thus A_{k+1} essentially lies between two random variables with mean $\frac{k}{m+1}$ and $\frac{k+1}{m+1}$, so to a good approximation $A_{k+1} \approx \frac{k}{m}$ for large m.

$$\frac{1}{(m-k)}\frac{(1-A_{k+1})}{f\left(F^{-1}(A_{k+1})\right)} \approx \frac{1-\frac{k}{m}}{(m-k)}\frac{1}{f\left(F^{-1}\left(\frac{k}{m}\right)\right)} = \frac{1}{mf\left(F^{-1}\left(\frac{k}{m}\right)\right)}.$$

As the multiple of an Exponential random variable is also an Exponential distribution with a suitably defined parameter, we to a very close approximation

$$Z_{(k+1)} - Z_{(k)} \sim \text{Exp}\left(\, m\, f\left(F^{-1}\left(\tfrac{k}{m}\right)\right)\,\right). \qquad \square$$

We use Lemma 4 to give the distribution of the spacings in three situations of interest, namely when the underlying distributions are Uniform, Exponential and Normal. The distribution of the original ACD samples $x_1, \ldots, x_\tau$, and hence random sums $S_1, \ldots, S_m$ when $l = 1$, are well-approximated by a Uniform distribution on $(0, 2^\gamma)$, In such a situation, the distribution of the consequent spacings has an Exponential distribution. More generally, the sum of $l > 1$ such distributions (Uniform or Exponential) is well-approximated by a Normal distribution even for moderate l, but the distribution of such a sum could always be calculated exactly if required using Lemma 4.

– **Uniform Distribution.** Suppose $Z_1, \ldots, Z_m \sim \text{Uni}(0, A)$ are uniformly distributed on $(0, A)$, then $Z_1, \ldots, Z_m$ have inverse distribution function $F^{-1}(u) = Au$ $(0 \leq$

$u \leq 1$) and density function $f(z) = A^{-1}$ ($0 \leq z < A$). Thus $f\left(F^{-1}(u)\right) = A^{-1}$, and the spacings have an Exponential distribution given by

$$Z_{(k+1)} - Z_{(k)} \sim \text{Exp}\left(mA^{-1}\right) \quad \text{with mean } \frac{A}{m}.$$

– **Exponential Distribution.** Suppose $Z_1, \ldots, Z_m \sim \text{Exp}(\lambda)$ are exponentially distributed with (rate) parameter λ (mean λ^{-1}), then $Z_1, \ldots, Z_m$ have inverse distribution function $F^{-1}(u) = -\lambda^{-1}\log(1-u)$ ($0 \leq u < 1$) and density function $f(z) = \lambda\exp(-\lambda z)$ ($z > 0$). Thus $f\left(F^{-1}(u)\right) = \lambda(1-u)$ ($0 \leq u < 1$), and the spacings have an Exponential distribution given by

$$Z_{(k+1)} - Z_{(k)} \sim \text{Exp}\left(\lambda(m-k)\right) \quad \text{with mean } \frac{1}{\lambda(m-k)}.$$

– **Normal Distribution.** Suppose $Z_1, \ldots, Z_m \sim \text{N}\left(\mu, \sigma^2\right)$ are normally distributed with mean μ and variance σ^2. If we let F^{-1} and f respectively denote the inverse distribution function and density function of such a $\text{N}(\mu, \sigma^2)$ random variable, then

$$f\left(F^{-1}(u)\right) = \frac{g\left(G^{-1}(u)\right)}{\sigma},$$

where G^{-1} and g are respectively the inverse distribution function and density function of a standard Normal $\text{N}(0,1)$ random variable. We therefore let $H(u)$ denote the function $g(G^{-1}(u))^{-1}$, so

$$H(u) = \frac{1}{g(G^{-1}(u))} = (2\pi)^{\frac{1}{2}}\exp\left(\texttt{InverseErfc}(2u)^2\right)\right) \quad [0 < u < 1],$$

where `InverseErfc` denotes the inverse function to the complementary error function, and we illustrate this function H in Figure 1. It can be seen that H is a moderately small value away from the extreme order statistics, for example $H(u) \approx 4$ for $0.2 < u < 0.8$. Thus the spacings have an Exponential distribution (with parameter depending on k) given by

$$Z_{(k+1)} - Z_{(k)} \sim \text{Exp}\left(\frac{m}{\sigma\ H\left(\frac{k}{m}\right)}\right) \quad \text{with mean } \frac{\sigma\ H\left(\frac{k}{m}\right)}{m}.$$

7.3 Aggressive shortening

Having shown that the sample amplification technique leads to relatively small spacings, we can now put everything together. The idea is to start with a list $\mathcal{L} = \{x_1, \ldots, x_\tau\}$ of ACD samples of mean value $2^{\gamma-1}$ and standard deviation $\sigma_0 \approx 3^{-\frac{1}{2}}2^{(\gamma-1)}$. One first amplifies this to a list of m samples S_k. One then sorts the S_k to get the order statistics $S_{(k)}$.[7] Compute the spacings $T_k = S_{(k+1)} - S_{(k)}$ for $k = 1, \ldots, m-1$ and store the

[7] In practice one can store the S_k in a binary search tree, in which case an explicit sorting step is not required.

$\tau = m/2$ "middle" spacings as input to the next iteration of the algorithm. After I iterations one then applies the orthogonal lattice attack.

We now analyse the method. The complexity is proportional to $Im\log(m)$, since each iteration computes a sorted list of size m. The mean and the standard deviation of the spacings is inversely proportional to m, so we would wish to take m to be very large. Suppose, at the j-th iteration, we have a list of τ_{j-1} values $Y_1^{(j-1)}, \ldots, Y_{\tau_{j-1}}^{(j-1)}$ (so $\tau_0 = \tau$) with standard deviation σ_{j-1}. As noted above, a random sum S_k is well-approximated as a Normal random variable with variance $l\sigma_{j-1}^2$ for $l > 1$. Lemma 4 shows that the k^{th} spacing in this Normal approximation case essentially has a distribution given by

$$S_{(k+1)} - S_{(k)} \sim \text{Exp}\left(\frac{m}{l^{\frac{1}{2}}\sigma_{j-1}\ H\left(\frac{k}{m}\right)}\right) \quad \text{with mean } \frac{l^{\frac{1}{2}}H\left(\frac{k}{m}\right)}{m}\sigma_{j-1}.$$

Figure 1 shows that $H(\frac{k}{m}) \approx 4$ when $0.2m \leq k \leq 0.8m$, so by considering the "middle" spacings of $T_1, \ldots, T_{m-1}$, we can obtain $\tau_j = \frac{1}{2}m$ random variables with approximately the same distribution that are in general independent. Thus at the end of the j^{th} iteration, we obtain random variables

$$Y_1^j, \ldots, Y_{\tau_j}^j \quad \text{with mean and standard deviation } \sigma_j = \frac{4l^{\frac{1}{2}}}{m}\sigma_{j-1}.$$

The main question is how many times the method can be iterated until the errors grow so large that p is not determined anymore. After j iterations, the random variables $Y_1^j, \ldots, Y_{\tau_j}^j$ are sums of $(2l)^j$ of the original ACD samples, so the standard deviation of an error term in the output of the j-th has increased by a multiple of $(2l)^{\frac{j}{2}}$. Hence, the total number of iterations performed satisfies $I < \eta$.

Our analysis shows that the average size of samples after i iterations is $(4\sqrt{l}/m)^i 2^{\gamma-1}$. To have samples of size close to η-bits thus requires

$$\eta \approx i\log_2(4\sqrt{l}/m) + \gamma - 1.$$

Hence, optimistically taking $i = \eta$, we need

$$\log_2(m) \approx (\gamma - 1 + \eta(\log(4\sqrt{l}) - 1)/\eta$$

In other words, the lists are of size close to $2^{\gamma/\eta}$, which is prohibitively large in practice. Even for the toy parameters $(\rho, \eta, \gamma) = (71, 2698, 19350000)$ from [CNT12] we would have $m \approx 2^{7000}$, which is absurd.

In summary, the detailed statistical analysis of this Section has essentially shown that a neighbouring difference approach, whilst initially appearing promising, can only reduce the essential magnitude and variability of the samples produced at each iteration by a factor that depends linearly on the number of sums considered at each iteration. For the parameter sizes required for a cryptographic system, this means that the resulting errors grow too rapidly for this approach to be useful.

It is natural to wonder why the BKW algorithm is a useful tool for LPN, and yet similar ideas are not useful for ACD. One answer is that ACD is actually a much easier problem than LPN: The naive attack on LPN takes 2^n operations, whereas one can solve ACD in vastly fewer than 2^γ steps.

8 Conclusions

We have surveyed known attacks on the ACD problem. Our main finding is that the multivariate polynomial attack is not more powerful than the orthogonal lattice attack, thereby clarifying the contribution of Cohn and Heninger [CH13]. We have developed a sample amplification method for ACD which may have applications in cryptanalysis. We have also investigated a pre-processing approach, similar to the BKW algorithm, and given a statistical analysis that explains why this method does not lead to an attack on ACD.

Acknowledgement

We thank Tancrède Lepoint and Nadia Heninger for valuable feedback on an earlier version of the the paper.

References

[Ajt06] M. Ajtai. Generating random lattices according to the invariant distribution. Preprint, 2006.

[BKW03] Avrim Blum, Adam Kalai and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. Journal of ACM, **50**, no. 4 (2003) 506−519.

[CN12] Y. Chen, P. Q. Nguyen. Faster algorithms for approximate common divisors: Breaking fully homomorphic encryption challenges over the integers. In D. Pointcheval and T. Johansson (eds.), EUROCRYPT'12, Springer LNCS7237 (2012) 502−519.

[CCK+13] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi and Aaram Yun. Batch Fully Homomorphic Encryption over the Integers. in T. Johansson and P. Q. Nguyen (eds.), EUROCRYPT 2013, Springer LNCS 7881 (2013) 315−335.

[CS15] J. H. Cheon, Damien Stehlé. Fully Homomorphic Encryption over the Integers Revisited. In E. Oswald and M. Fischlin (eds.), EUROCRYPT'15, Springer LNCS 9056 (2015) 513−536.

[CH13] H. Cohn, N. Heninger. Approximate common divisors via lattices. in proceedings of ANTS X, vol. 1 of The Open Book Series, pages 271−293, 2013.

[CMNT11] J.-S. Coron, A. Mandal, D. Naccache, M. Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In P. Rogaway (ed.), CRYPTO 2011, Springer LNCS 6841 (2011) 487−504.

[CNT12] J.-S. Coron, D. Naccache, M. Tibouchi. Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers. In D. Pointcheval and T. Johansson (eds.), EUROCRYPT'12, Springer LNCS 7237 (2012) 446−464.

[DGHV10] M. van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan. Fully homomorphic encryption over the integers. In H. Gilbert (ed.), EUROCRYPT 2010, Springer LNCS 6110 (2010) 24−43.

[DT14] J. Ding, C. Tao. A New Algorithm for Solving the General Approximate Common Divisors Problem and Cryptanalysis of the FHE Based on the GACD problem. Cryptology ePrint Archive, Report 2014/042, 2014.

[Geb16] S. W. Gebregiyorgis, Algorithms for the Elliptic Curve Discrete Logarithm and the Approximate Common Divisor Problem, PhD Thesis, University of Auckland, 2016.

[HG01] N. Howgrave-Graham. Approximate integer common divisors. in J. Silverman (ed), Cryptography and Lattices, Springer LNCS 2146 (2001) 51−66.

[LS14] Hyung Tae Lee and Jae Hong Seo. Security Analysis of Multilinear Maps over the Integers. in CRYPTO 2014, Springer LNCS 8616 (2014) 224−240.

[LLL82] A. Lenstra, H. W. Lenstra Jr., L. Lovász. Factoring polynomials with rational coefficients. Math. Ann., Vol. 261, No. 4 (1982) 515−534.

[Lep14] T. Lepoint, Design and implementation of lattice-based cryptography, PhD thesis, LUniversité Du Luxembourg and LÉcole Normale Supérieure (2014)

[Lyu05] V. Lyubashevsky. The Parity Problem in the Presence of Noise, Decoding Random Linear Codes, and the Subset Sum Problem. APPROX-RANDOM 2005, Springer LNCS 3624 (2005) 378−389.

[NgSt06] Phong Q. Nguyen and Damien Stehlé. LLL on the Average. In Florian Hess, Sebastian Pauli and Michael E. Pohst (eds.), ANTS-VII, Springer LNCS 4076 (2006) 238−256.

[NgSt01] Phong Q. Nguyen and Jacques Stern. The Two Faces of Lattices in Cryptology. In J. Silverman (ed), Cryptography and Lattices, Springer LNCS 2146 (2001) 146−180.

[Pyk65] R. Pyke. Spacings. Journal of the Royal Statistical Society Series B, volume 27 (1965) 395−449.