Submitted exclusively to the London Mathematical Society
doi:10.1112/0000/000000

Algorithms for the Approximate Common Divisor Problem

Steven D. Galbraith, Shishay W. Gebregiyorgis and Sean Murphy

Abstract

The security of several homomorphic encryption schemes depends on the hardness of variants of the Approximate Common Divisor (ACD) problem. We survey and compare a number of lattice-based algorithms for the ACD problem, with particular attention to some very recently proposed variants of the ACD problem. One of our main goals is to compare the multivariate polynomial approach with other methods. We find that the multivariate polynomial approach is not better than the orthogonal lattice algorithm for practical cryptanalysis.

We also briefly discuss a sample-amplification technique for ACD samples and a pre-processing algorithm similar to the Blum-Kalai-Wasserman (BKW) algorithm for learning parity with noise. The details of this work are given in the full version of the paper.

1. *Introduction*

The approximate common divisor problem (ACD) was first studied by Howgrave-Graham [**13**]. Further interest in this problem was provided by the homomorphic encryption scheme of van Dijk, Gentry, Halevi and Vaikuntanathan [**9**] and its variants [**7**, **8**, **5**]. 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 integer, the x_i are γ bits, and the r_i are ρ bits, where ρ is significantly smaller than η. For a precise definition see Section 2.

The original papers [**13**, **9**] sketched a large number of possible lattice attacks on this problem, including using orthogonal lattices and Coppersmith's method. (For background on lattices see Appendix A.) Further cryptanalytic work was done by [**3**, **6**, **8**, **10**]. Our paper surveys and compares the known lattice algorithms for the ACD problem. We study several recently proposed variants of the ACD problem [**4**, **16**, **5**], and argue that they may offer greater security than the original ACD proposal. Our main finding is that the orthogonal lattice approach is better than the multivariate polynomial approach for practical cryptanalysis.

We also briefly discuss 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). The details can be found in the full version of the paper [**11**].

We do not consider in this paper the variants of "exhaustive search" over the errors r_i, as proposed by Chen and Nguyen [**3**], Coron, Naccache and Tibouchi [**8**], and Lee and Seo [**14**].

2. *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 integer, so that

$$2^{\eta-1} < p < 2^{\eta}. \tag{2.1}$$

2000 *Mathematics Subject Classification* 11Y16 (primary), 94A60 (secondary).

It is not necessary for p to be prime, and in some applications (e.g., Appendix D of [**9**]) it is definitely not prime. Define the efficiently sampleable distribution $\mathcal{D}_{\gamma,\rho}(p)$ as

$$\mathcal{D}_{\gamma,\rho}(p) = \{pq + r \mid q \leftarrow \mathbb{Z} \cap [0, 2^\gamma/p), r \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)\}. \tag{2.2}$$

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

$$2^{\gamma-1} < x_i < 2^\gamma. \tag{2.3}$$

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 (see [**9**]) 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.

Suggested parameters in [**9**] are $(\rho, \eta, \gamma) = (\lambda, \lambda^2, \lambda^5)$, where λ is a security parameter, so one sees that ρ is extremely small compared with η.

Cheon et al [**4**] 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 $x_0 = \pi q_0$, where $\gamma \geq \ell\eta$. A ciphertext is an element $c = \pi q + r$ where r is congruent modulo each prime p_j to a small integer r_j, 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. For later use we introduce the notation $x_i = p_j q_{i,j} + r_{i,j}$ for $1 \leq i \leq t$ and $1 \leq j \leq \ell$. We refer to [**4**] and Chapter 7 of Lepoint [**16**] 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 [**4**] 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 [**5**] 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 [**5**] 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 [**5**]: 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. In particular, the Cheon-Stehlé-ACD seems to offer a higher degree of security, which is not surprising since that problem enjoys some evidence for its hardness.

3. *Simultaneous Diophantine approximation approach (SDA)*

This section recalls the simplest, and still one of the most effective, lattice attacks on ACD. It was first described by Howgrave-Graham (see Section 2 of [**13**]) and was further developed in Section 5.2 of [**9**]. For the benefit of non-expert readers we present all the details.

The basic idea of this attack is to note that if $x_i = pq_i + r_i$ for $0 \le i \le t$, where r_i is small, then

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

for $1 \le i \le t$. In other words, the fractions q_i/q_0 are an instance of simultaneous Diophantine approximation to x_i/x_0. If one can determine such a fraction q_i/q_0 then one can solve the ACD problem by computing $r_0 \equiv x_0 \pmod{q_0}$ and hence $(x_0 - r_0)/q_0 = p$. We will see 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 [**9**] we build a lattice L of rank $t+1$ generated by the rows of the basis matrix

$$\mathbf{B} = \begin{pmatrix} 2^{\rho+1} & x_1 & x_2 & \cdots & x_t \\ & -x_0 & & & \\ & & -x_0 & & \\ & & & \ddots & \\ & & & & -x_0 \end{pmatrix}. \tag{3.1}$$

Note that L contains the vector

$$\begin{aligned} \mathbf{v} &= (q_0, q_1, \cdots, q_t)\mathbf{B} \\ &= (q_0 2^{\rho+1}, q_0 r_1 - q_1 r_0, \cdots, q_0 r_t - q_t r_0) \end{aligned}$$

Since $q_i \approx 2^{\gamma-\eta}$ the Euclidean norm of $\mathbf{v}$ is approximately $\sqrt{t+1}\, 2^{\gamma-\eta+\rho+1}$ (we give a more precise estimate in Lemma 3.1). 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 solution $\mathbf{w}$ for the shortest non-zero vector. One then divides the first entry of $\mathbf{w}$ by $2^{\rho+1}$ to get a candidate value for q_0 and then computes $r_0 = x_0 \pmod{q_0}$ and $p = (x_0 - r_0)/q_0$. One can then "test" this value for p by checking if $x_i \pmod{p}$ are small for all $1 \le i \le t$. We call this the **SDA algorithm**.

3.1. *Heuristic analysis of the SDA Algorithm*

The method is analysed in Section 5.2 of [**9**], 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 argued that we need $t > \gamma/\rho$ to have any chance for this method to succeed, even disregarding the difficulties of lattice reduction methods to find the shortest vector.

We now refine the analysis of [**9**], with an eye to the Cheon-Stehlé-ACD parameters. First we state and prove Lemma 3.1, which gives an improved estimate of the size of the target vector.

LEMMA 3.1. *The expected length of the target vector* $\mathbf{v}$ *has a tight upper bound of*

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

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

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

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

$$\begin{aligned}\mathbf{E}\left((q_0r_i - q_ir_0)^2\right) &= \mathbf{E}\left(q_0^2r_i^2\right) + \mathbf{E}\left(q_i^2r_0^2\right) - 2\mathbf{E}\left(q_0r_iq_ir_0\right)\\ &= \mathbf{E}\left(q_0^2\right)\mathbf{E}\left(r_i^2\right) + \mathbf{E}\left(q_i^2\right)\mathbf{E}\left(r_0^2\right) - 2\mathbf{E}\left(q_0q_i\right)\mathbf{E}\left(r_i\right)\mathbf{E}\left(r_0\right)\\ &\approx \tfrac{2}{9}p^{-2}2^{2(\rho+\gamma)}.\end{aligned}$$

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

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

Jensen's Inequality shows that $\mathbf{E}(|v|) \leq \mathbf{E}(|v|^2)^{\frac{1}{2}}$, and this bound is tight for large dimension t. This completes the proof. □

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 we replace this with heuristics. Hence, using equation (A.2), the target vector is the shortest vector in the lattice and is found by LLL if

$$0.47\sqrt{t+1}(1.04)^{t+1}2^{\gamma+\rho-\eta} < \sqrt{(t+1)/(2\pi e)}2^{(\rho+1+\gamma t)/(t+1)}. \tag{3.2}$$

Ignoring constants and the $(1.04)^{t+1}$ term in the above equation, we find that a necessary (not sufficient) condition for the algorithm to succeed is

$$t+1 > \frac{\gamma-\rho}{\eta-\rho}. \tag{3.3}$$

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. This suggests the Cheon-Stehlé variant of ACD is significantly harder than the original variant from [**9**], which is consistent with the fact that [**5**] contains a reduction from their variant of ACD to the LWE problem.

The above analysis ignored some terms, we refer to the full version of the paper [**11**] for a more careful analysis.

It is interesting to consider this attack in the context of the CRT-ACD. In this variant we have $x_i = p_jq_{i,j} + r_{i,j}$ for $1 \leq j \leq \ell$ where each $r_{i,j}$ is small. It follows that the lattice contains the vectors

$$(q_{0,j}2^{\rho+1}, q_{0,j}r_{1,j} - q_{1,j}r_{0,j}, \cdots, q_{0,j}r_{t,j} - q_{t,j}r_{0,j})$$

for all $1 \leq j \leq \ell$ and these all have similar length. It is important to note that any one of these vectors allows to compute one of the prime factors p_j, by computing $q_{0,j}$ from the first component of the vector and then $p_j = \lfloor x_0/q_{0,j} \rceil$. But if $(u_1 2^{\rho+1}, u_2, \ldots, u_t)$ is a short linear combination of several of these vectors then there is no reason to expect $x_0 \pmod{u_1}$ to be a small integer, or that $\lfloor x_0/u_1 \rfloor$ is one of the primes in the private key.

4. *Orthogonal based approach (OL)*

Nguyen and Stern (see for example [**19**]) 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 [**9**] 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 [**9**] 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 [**10**] have given a method based on vectors orthogonal to $(q_1, \ldots, q_t)$. Cheon and Stehlé [**5**] have considered the second method from Appendix B.1 of [**9**].

Our analysis (as with that in [**9**]) and experiments suggest all these methods have essentially the same performance in both theory and practice. Indeed, all three methods end up computing short vectors that are orthogonal to both the vector $(q_1, \ldots, q_t)$ and some vector related to the error terms r_i, for example see Lemma 4.2. Hence, in this paper we follow [**9**, **5**] and study the use of vectors orthogonal to $(1, -r_1/R, \ldots, -r_t/R)$. The 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

$$\mathbf{B} = \begin{pmatrix} x_1 & R & & & & & \\ x_2 & & R & & & & \\ x_3 & & & R & & & \\ \vdots & & & & \ddots & & \\ x_t & & & & & & R \end{pmatrix}. \tag{4.1}$$

Clearly the rank of B is t. The lattice volume was estimated in previous works, but we give an exact formula (the proof is given in the full version of the paper [**11**]).

LEMMA 4.1. *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 $R^{t-1}\sqrt{R^2 + x_1^2 + \cdots + x_t^2}$.*

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. [**9**] is

$$v_0 - \sum_{i=1}^{t} \frac{v_i}{R} r_i = \sum_{i=1}^{t} u_i x_i - \sum_{i=1}^{t} \frac{u_i R}{R} r_i = \sum_{i=1}^{t} u_i (x_i - r_i) = 0 \pmod{p}. \tag{4.2}$$

Since we don't know p, we wish to have a linear equation over $\mathbb{Z}$. The equality holds over $\mathbb{Z}$ 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 4.2. *Let* $\mathbf{B}$ *be as in equation (4.1). 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*

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

Proof. Let $\mathbf{v} = (v_0, v_1, \cdots, v_t) = \left(\sum_{i=1}^{t} u_i x_i, u_1 R, u_2 R, \cdots, u_t R\right)$ 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

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

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. Both [**9**, **5**] 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**.

Following the analysis in [**9**, **5**], if one ignores constants and assumes the lattice reduction algorithm is perfect then a necessary condition on the dimension is $t \geq (\gamma - \rho)/(\eta - \rho)$, which is the same as equation (3.3) for the SDA method. Hence, we deduce that the OL method has essentially the same power as the SDA method. Our experimental results (see Table B.2) 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). For more details about the analysis we refer to the full version of the paper [**11**], and also [**9**, **5**].

5. *Multivariate polynomial approach (MP)*

Howgrave-Graham [**13**] 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 extended in Appendix B.2 of van Dijk et al [**9**]. Finally, a detailed analysis was given by Cohn and Heninger [**6**]. This approach has some advantages if the number of ACD samples is very small (the original context studied in [**13**]), but we focus on the use of this method in practical cryptanalysis where the number of samples is large. Our heuristic analysis and experimental results suggest that, assuming sufficiently many ACD samples are available, 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.

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

We change notation from the rest of the paper to follow more closely the notation used in [**6**]. 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 [**6**], such a multivariate polynomial is constructed as integer 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 $t \geq 1$ (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 defined by the coefficient row vectors of the polynomials

$$f_{[i_1,\ldots,i_m]}(X_1, \ldots, X_m) = (RX_1 - a_1)^{i_1} \cdots (RX_m - a_m)^{i_m} N^{\ell}, \tag{5.1}$$

such that $i_1 + \cdots + i_m \leq t$ and $\ell = \max(k - \sum_j i_j, 0)$. It is shown in [**6**] that L has dimension $d = \binom{t+m}{m}$ and determinant

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

where we use the natural choice $R = 2^{\rho}$. There is no benefit to taking $k > t$, as it leads to the entire matrix for the case $t = k$ being multiplied by the scalar N^{k-t}.

Let $\mathbf{v}$ be a vector in L. One can interpret $\mathbf{v} = (v_{i_1,\cdots,i_m} R^{i_1+\cdots+i_m})$ as the coefficient vector of a polynomial

$$Q(X_1, \ldots, X_m) = \sum_{i_1,\cdots,i_m} v_{i_1,\cdots,i_m} X_1^{i_1} \cdots X_m^{i_m}.$$

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

$$\begin{aligned} |Q(r_1, \cdots, r_m)| &\leq \sum_{i_1,\cdots,i_m} |v_{i_1\cdots i_m}| |r_1|^{i_1} \cdots |r_m|^{i_m} \\ &\leq \sum_{i_1,\cdots,i_m} |v_{i_1\cdots i_m}| R^{i_1} \cdots R^{i_m} \\ &= \|\mathbf{v}\|_1. \end{aligned}$$

Hence, if $\|\mathbf{v}\|_1 < p^k$ then we have an integer polynomial with the desired root. 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**.

Note that the case $(t, k) = (1, 1)$ gives essentially the same lattice as in equation (4.1) and so this case of the MP algorithm is the same as the orthogonal lattice attack (this was already noted in [**9**]). Our concern is whether taking $t > 1$ gives rise to a better attack. We will argue

that, when the number of ACD samples is large, the best choices for the MP algorithm are $(t, k) = (1, 1)$, and so the MP method seems to have no advantage over the orthogonal lattice method.

5.1. *Heuristic Analysis*

Cohn and Heninger [**6**] give a heuristic theoretical analysis of the MP algorithm and suggest optimal parameter choices (t, m, k). We briefly recall the ideas, and refer to [**6**] for details. 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".

The MP algorithm succeeds if it produces m vectors in the lattice such that $\|\mathbf{v}\|_1 < p^k$. Using $\|\mathbf{v}\|_1 \leq \sqrt{d}\|\mathbf{v}\|$ (where the latter is the Euclidean norm) and the bounds from Assumption A.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

$$d\log_2(d) + d^2\log_2(1.02) + d\rho\frac{mt}{m+1} + \gamma\binom{k+m}{m}\frac{k}{m+1} < k\eta d. \tag{5.2}$$

Cohn and Heninger [**6**] introduce a parameter $\beta = \eta/\gamma \ll 1$ so that $p \approx N^\beta$. They work with the equation

$$\frac{mt\rho}{(m+1)k} + \frac{\gamma k^m}{(m+1)t^m} < \eta = \beta\gamma \tag{5.3}$$

which is a version of equation (5.2), with some terms deleted and approximating $\binom{k+m}{m} \approx k^m$ and $d = \binom{t+m}{m} \approx t^m$.

Their analysis assumes m is fixed and considers taking t large. They choose $t \approx \beta^{-1/m}k$, which means that $t \gg k$. Their method allows errors up to $R = 2^\rho = N^{\beta^{(m+1)/m}}$.

The main "heuristic theorem" of [**6**] 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.

Note that the dimension in their method is approximated as $t^m \approx \beta^{-1}k^m \geq \beta^{-1} = \gamma/\eta$, so we yet again we encounter the same dimension bound as the previous methods (at least, when ρ is small).

5.2. *Further Analysis*

We now consider the parameters more generally, unlike in [**6**] where it was assumed that the optimal solution would be to take $t, k > 1$.

Section 2.1 of [**6**] suggests $k \approx (\beta^{1+\epsilon}\log_2(N))^{1/(2\epsilon m)}$. Taking $m \to \infty$ with these parameters results in $(t, k) = (1, 1)$, which is consistent with our claim that $(t, k) = (1, 1)$ is optimal when m may be chosen to be large. However, one could speculate that a general analysis could lead to different asymptotics. So we give a more general analysis.

We now derive some useful necessary conditions from equation (5.2) for the algorithm to succeed. Noting that, for large m, $\frac{mt}{m+1} \approx t$ we see that it is necessary to have

$$t\rho < k\eta, \tag{5.4}$$

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

$$d = \binom{t+m}{m} > \frac{\gamma}{\eta}\binom{k+m}{m}\frac{1}{m+1}. \tag{5.5}$$

When $k = 1$ then the right hand side is equal to γ/η, but it gets steadily larger as k grows. This provides some evidence that $k > 1$ may not lead to an optimal attack. The bound also shows that the MP method does not overcome the minimal degree bound γ/η we already saw for the SDA and OL methods, at least when ρ is small. (In the case $(t, k) = (1, 1)$ equation (5.2) essentially becomes $d + 1 > \gamma/(\eta - \rho)$ which we have already seen in Sections 4 and 3.)

More generally, equation (5.2) implies, when m is large,

$$d\rho t \ + \ \gamma\binom{k+m}{m}\tfrac{k}{m+1} < k\eta d.$$

Dividing by k and re-arranging gives

$$d > \frac{\gamma}{\eta - \frac{t}{k}\rho}\binom{k+m}{m}\frac{1}{m+1}.$$

Since $\frac{t}{k} \geq 1$ and $\binom{k+m}{m}\frac{1}{m+1} \geq 1$ we see that this is never better than the lattice dimension bound $d > \frac{\gamma}{\eta-\rho}$ from equation (3.3). Hence, there seems no theoretical reason why, when m is large, the MP method should be better than the SDA or OL methods. Our practical experiments confirm this (see below).

5.3. *Comments*

A further major advantage of the SDA and OL methods compared with the MP approach with $t > 1$ 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.

Note that the heuristics 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.

Table A.1 at the end of the paper gives a comparison of different parameters for the MP method with $\eta = 100$ 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 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. Table B.1 considers larger values for γ (still with $\eta = 100$) and the aim of this table is to emphasise the considerable increase in the running time when using $t > 1$.

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 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 $\binom{k+m}{m} \approx d = \binom{t+m}{m}$. In the case $t = k$, dividing equation (5.2) by dt implies $d \geq m + 1 > \gamma/(\eta - \rho)$. Again, this suggests the MP approach has no advantage over other methods for parameters of this type. Our experimental results confirm this (see Table A.1).

6. *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 [**2**] 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.

6.1. *Preserving the sample size*

Suppose $B = 2^b$. After k iterations we have generated approximately $\tau - kB$ samples, each of $\gamma - kb$ 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

[†]At first glance this approach may not seem to have any advantage over directly forming a lattice from the samples. But we stress that this is not the case. Imagine that one has $\tau > 10^6$ ACD samples. One would not work with a lattice of dimension greater than a million. Instead the idea is to construct a smaller list of "better quality" samples from the original ones, and then solve a lattice problem corresponding to the smaller set of samples.

$\frac{1}{3}2^{2\rho+k}$. 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. Hence, an absolute upper limit on the number of iterations is $2(\eta - \rho)$. After that many iterations the samples are reduced to around $\gamma - 2b(\eta - \rho)$ 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 k iterations of pre-processing we would need a lattice of dimension

$$\frac{\gamma - bk}{\eta - (\rho + k/2)}.$$

Even in the best possible case when one can take $k = 2(\eta - \rho)$ and keep the denominator constant at $(\eta - \rho)$, we see that the lattice dimension is lowered from γ/η to $(\gamma/\eta) - 2b$. Since a typical value for b is 8 or 16, this approach can make very little difference to the problem.

6.2. *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. Precisely, if $\{x_1, \ldots, x_\tau\}$ is a list of ACD samples, then we can generate m new ACD samples as

$$S_k = \sum_{i \in I_k} x_i \qquad [k = 1, \ldots, m],$$

where $I_k \subseteq \{1, \ldots, \tau\}$ are randomly chosen sets of size l (small). This is similar to ideas used to amplify the number of samples for solving LPN or LWE [**17**].

The idea is then to sort these new samples as $S_{(1)} \leq \cdots \leq S_{(m)}$ and then take successive differences or *spacings* $T_k = S_{(k+1)} - S_{(k)}$ for $k = 1, \ldots, m-1$. The statistical distribution of spacings arising from a general distribution is considered by Pyke [**20**]. The full version of the paper [**11**] describes these statistics in detail and analyses the resulting algorithm.

Our analysis shows 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.

Acknowledgements

We thank the anonymous referees for their careful reading of the paper, and Nadia Heninger for insights into [**6**].

References

1. M. Ajtai. Generating random lattices according to the invariant distribution. Preprint, 2006.
2. 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.
3. 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.
4. Jung Hee Cheon, Jean-Sbastien Coron, Jinsu Kim, Moon Sung Lee, Tancrde 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.
5. J. 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.
6. 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.
7. J. 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.
8. J. 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.
9. 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.
10. 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.
11. Steven D. Galbraith, Shishay W. Gebregiyorgis and Sean Murphy, Algorithms for the Approximate Common Divisor Problem, eprint 2016/215 (2015)
12. S. W. Gebregiyorgis, Algorithms for the Elliptic Curve Discrete Logarithm and the Approximate Common Divisor Problem, PhD Thesis, University of Auckland, 2016.
13. N. Howgrave-Graham. Approximate integer common divisors. in J. Silverman (ed), Cryptography and Lattices, Springer LNCS 2146 (2001) 51−66.
14. H. T. Lee and J. H. Seo. Security Analysis of Multilinear Maps over the Integers. In CRYPTO 2014, Springer LNCS 8616 (2014) 224−240.
15. A. Lenstra, H. Lenstra, L. Lovász. Factoring polynomials with rational coefficients. Math. Ann., Vol. 261, No. 4 (1982) 515−534.
16. T. Lepoint, Design and implementation of lattice-based cryptography, PhD thesis, LUniversité Du Luxembourg and LÉcole Normale Supérieure (2014)
17. 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.
18. 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.
19. 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.
20. R. Pyke. Spacings. Journal of the Royal Statistical Society Series B, volume 27 (1965) 395−449.

Appendix A. *Lattice basis reduction*

The algorithms considered in this paper make use of lattice basis reduction algorithms such as LLL [**15**]. 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 [**1**] (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 [**1**] 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

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

for $1 \leq i \leq n$.

It is folklore that LLL performs better on average than these worst-case bounds suggest. Nguyen and Stehlé [**18**] have studied the behaviour of LLL on "random" lattices and have hypothesised that an LLL-reduced basis satisfies the improved bound

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

By analogy with the relationship between the worst-case bounds $\|\mathbf{b}_1\| < 2^{n/4}\det(L)^{1/n}$ and $\|\mathbf{b}_1\| < 2^{n/2}\lambda_1(L)$ it is natural to suppose that

$$\|\mathbf{b}_1\| \leq (1.04)^n \lambda_1(L). \tag{A.2}$$

Figure 4 of [**18**] 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 [**9**, **5**] is under an assumption of this type. We formalise this with the below heuristic assumption.

Assumption A.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 $1 \leq i \leq n$.

In practice one uses lattice reduction algorithms such as BKZ that give better approximation factors. But the above heuristics are sufficient for our analysis.

Appendix B. *Experimental Results*

[TO BE SHORTENED IF THE CHAIRS REQUIRE]

We have conducted extensive experiments with the SDA, OL and MP methods. For a brief comparison of the SDA and OL see Table B.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 [**12**].

The parameters (ρ, η, γ) in Table B.2 are selected according to the formula $(\lambda, \lambda + d\log(\lambda), d^2\lambda\log(\lambda))$ from [**5**], 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 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.

Steven D. Galbraith and Shishay W. Gebregiyorgis
Mathematics Department,
University of Auckland,
New Zealand.

S.Galbraith@math.auckland.ac.nz
sgeb522@aucklanduni.ac.nz
Sean Murphy
Royal Holloway, University of London, UK.

S.Murphy@rhul.ac.uk

TABLE A.1. *Comparison between different parameter choices (t, k) in the multivariate polynomial (MP) algorithm. We have $\eta = 100$ and list the largest value for ρ (denoted $\rho_{\max}$) that can be solved with reasonable probability for the given choice (γ, η, t, k, m).* $\dim(L)$, 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.*

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

TABLE B.1. *Comparison between different parameter choices (t, k) in the multivariate polynomial algorithm with $\eta = 100$. $\dim(L)$, 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	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	**

TABLE B.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