Japan Journal of Industrial and Applied Mathematics (2018) 35:1123–1152
https://doi.org/10.1007/s13160-018-0316-x

ORIGINAL PAPER

Area 2

Cryptanalysis of a public key cryptosystem based on Diophantine equations via weighted LLL reduction

Jintai Ding[1] · Momonari Kudo[2,3,4] · Shinya Okumura[5,6] · Tsuyoshi Takagi[6,7] · Chengdong Tao[8]

Received: 21 December 2016 / Revised: 31 May 2018 / Published online: 21 June 2018

Abstract
Researching post-quantum cryptography is now an important task in cryptography. Although various candidates of post-quantum cryptosystems (PQC) have been constructed, sizes of their public keys are large. Okumura constructed a candidate of PQC whose security is expected to be based on certain Diophantine equations (DEC). Okumura analysis suggests that DEC achieves the high security with small public key sizes. This paper proposes a polynomial time-attack on the one-way property of DEC. We reduce the security of DEC to finding special short lattice points of some low-rank lattices derived from public data. The usual LLL algorithm could not find the most important lattice point in our experiments because of certain properties of the lattice point. Our heuristic analysis leads us to using a variant of the LLL algorithm, called a weighted LLL algorithm by us. Our experiments suggest that DEC with 128 bit security becomes insecure by our attack.

Keywords Weighted LLL reduction · Public-key cryptosystem · Post-quantum cryptosystem · Diophantine equation

Mathematics Subject Classification 94A60 · 11Y16

1 Introduction

Researching post-quantum cryptography is now an important task in cryptography. In fact, National Institute of Standards and Technology published a draft of the report on post-quantum cryptography NISTIR 8105 [23] (see also their announcement at PQCrypto 2016 [24]). Although various cryptosystems expected to be post-quantum cryptosystems (PQC) have been already constructed, see [7,11] for details, sizes of their public keys are large. Thus finding computationally-hard problems which allow us to construct PQC with public keys of small sizes is a very important task in cryptography.

Extended author information available on the last page of the article

A Diophantine problem is well-known to be a computationally-hard problem in mathematics [12], and there are some cryptographic schemes based on the problem [6,17,20,31], which are expected to have resistance to quantum algorithms. (Note that Diophantine problem here means a problem to find integral or rational zeros of a given multivariate polynomial with integer coefficients and high degree.) However, a polynomial time-attack on the one-way property of the cryptosystem [20] is proposed [10], and Proposition 2 in [17] suggests that the protocols [6,17,31] are impractical.

We can also consider the Diophantine problems over other rings. The Algebraic Surface Cryptosystem (ASC) [4] is based on the difficulty of the section finding problem, which can be viewed as the Diophantine problem over global function fields. Such the Diophantine problem is shown to be unsolvable in general [26,29]. The security analysis suggests that ASC with public keys of sizes of about 500 bits achieves high security, see [4]. However, the ideal decomposition attack [15] breaks the one-way property of ASC.

Okumura [25] constructed a candidate of PQC of which the security is expected to be based on the difficulty of solving a special class of Diophantine equations, called Diophantine equations of degree increasing type, over $\mathbb{Z}$ (we will recall a definition of a polynomial of degree increasing type in Sect. 3). We call this cryptosystem DEC for short. Okumura shows that the solvability of Diophantine equations of degree increasing type is undecidable in general, see Remark 3.2 of [25]. DEC is a number field analogue of ASC and use the twisted plaintext, obtained from a plaintext by using RSA-like modular arithmetic, and some random polynomials with large coefficients in the encryption process. These are the main ideas of DEC to resist the analogues of all attacks [15,18,28,30] on ASC and cryptosystems [1–3], which are proposed previously as ASC. In Sect. 4 of [25], Okumura points out that the above ideas increase the number of possible parameters in DEC, and that breaking the one-way property of DEC will become infeasible. Okumura also points out that one can decode a plaintext correctly from the twisted plaintext by using polynomials of degree increasing type as public keys. We will review DEC and its recommended parameters briefly in Sect. 3.

Another important property of DEC in post-quantum cryptography is that we may use public keys with small sizes, e.g., about 1,200 bits with 128 bit security (see Remark 9). The size (1,200 bits) is about 10 times smaller than sizes of public keys used in cryptosystems [21,22,27], which are well-known to be efficient among the candidates of PQC, with 128 bit security. Thus we consider that the security analysis of DEC is an important task in cryptography.

1.1 Our contribution

In this paper, we propose a polynomial time-attack on DEC. We show a linearization technique to transform the one-way property of DEC to finding appropriate solutions of linear systems obtained from public data. The use of three polynomials as a ciphertext enables us to use the linearization technique which constructs linear systems. This is the first weakness of DEC. Our attack consists of three steps. In each step, we have a linear system and need to find its appropriate solution, i.e., we need to find an appropriate lattice point in the lattice which is the solution space of the linear system. We use a solution obtained in the first (resp. second) step to construct a linear system in

the second (resp. third) step. After finding appropriate solutions of the linear systems in all the steps, it is possible to recover a plaintext with sufficiently high probability by applying the Babai nearest plane algorithm [5] and some modular arithmetic.

Our various experiments on our attack in Sect. 6 suggest that finding a correct solution results in breaking DEC with sufficiently high probability. More precisely, after we find a correct solution in the first step, we can solve the linear systems in the second and third steps (note that in the third step, we may use an incorrect solution obtained in the second step). Thus the success of the first step is most important for our attack.

The rank of the lattice occurring in the first step is low, e.g., 3-rank in almost all cases, and a target lattice point in the first step is relatively short in the lattice. The quality of basis reduction algorithms such as the LLL algorithm [19] depends heavily on the rank of a lattice, and the LLL algorithm outputs a shortest lattice point in many cases for 3-rank lattices, see [19]. Thus it seems that one can succeed in the first step by using the LLL algorithm (or other basis reduction algorithms). However, as we will see in Sect. 4.3, the usual LLL algorithm does not seem to work well for finding the target lattice point in the first step, where by the "usual LLL algorithm", we mean the LLL algorithm in terms of p-norms ($1 \leq p \leq \infty$) $\|\boldsymbol{a}\|_p := (|a_1|^p + \ldots + |a_n|^p)^{\frac{1}{p}}$. We heuristically analyse a reason why the usual LLL algorithm is not useful in our attack as follows: the target lattice point in the first step is not shortest, in terms of p-norms ($1 \leq p \leq \infty$), with high probability, but some of its entries are comparatively small. In other words, the target lattice point is a comparatively short (not neccessarily shortest) in terms of well-known norms and has entries of unbalanced sizes.

1.2 Weighted LLL

In order to find lattice points having such properties, we find a special norm which makes the target lattice point in the first step (nearly) shortest by a heuristic way and apply a special LLL algorithm in terms of the special norm. We call the special norm and the special LLL algorithm the *weighted norm* and the *weighted LLL algorithm*, respectively. By a weighted norm for a vector $\boldsymbol{a} = (a_1, ..., a_n)$, we mean the norm:

$$\|\boldsymbol{a}\| = \sqrt{(a_1 w_1)^2 + \cdots + (a_n w_n)^2},$$

where w_i's are positive real numbers, which we call the weight factors. Note that as we already mentioned above, using other well-known norms, e.g., the p-norms ($1 \leq p \leq \infty$), in the LLL algorithm does not seem to be effective in finding the target lattice point.

We also note that using the weighted LLL algorithm can be also considered as using a re-scaling of a lattice to find lattice points with entries of unbalanced sizes in an LLL reduced basis of the lattice. Such a method can be also found in Coppersmith's method [9] (see also Chapter 19 of [16]) and in Faugére et al.'s method [14]. In our method, each entries of the weighted norm are 2-power integers to use the knowledge of the bit length of entries of our target lattice point as in Faugére et al.'s method [14] (the possibility of knowing the bit length of entries of our target lattice point is the second weakness of DEC).

1.3 Experimental verification of our attack

Our many experiments in Sect. 6 suggest that the weighted LLL algorithm can find target lattice points in the first step of our attack with high probability (the probability being about from 70 to 90%) for the recommended parameters in Sect. 3. These results suggest that the weighted LLL algorithm is effective in cryptanalysis of cryptosystems whose security are reduced to finding lattice points with special properties: they are not shortest, but the bit length of their entries are almost known and comparatively small among entries of lattice points in certain lattices. In addition, our experiments also suggest that our attack breaks the one-way property of DEC with probability being about from 20 to 40% (this probability is sufficient in practical cryptanalysis). Our detailed complexity analysis on our attack and our experiments show that our attack is performed in polynomial time, and thus we conclude that our attack via the weighted LLL algorithm is practical and makes DEC insecure.

This paper is organized as follows: In Sect. 2, we give a definition of a weighted norm and describe the weighted LLL algorithm. In Sect. 3, we give a brief review of DEC. In Sect. 4, we describe the outline and some assumptions of our attack, and we also give an algorithm of our attack and a toy example to illustrate our attack. In Sect. 5, we analyse the complexity on our attack. In Sect. 6, we give some experimental results on our attack.

Notation

Throughout this paper, we denote by $R[\underline{x}] := R[x_1, \ldots, x_n]$ the polynomial ring with n variables over a ring R. For every $\underline{\boldsymbol{i}} = (i_1, \ldots, i_n) \in \left(\mathbb{Z}_{\geq 0}\right)^n$ and $\underline{\boldsymbol{a}} = (a_1, \ldots, a_n) \in R^n$, we denote the element $a_1^{i_1} \cdots a_n^{i_n} \in R$, the monomial $x_1^{i_1} \cdots x_n^{i_n} \in R[\underline{x}]$ and the value $\sum_{k=1}^n i_k$ by $\underline{\boldsymbol{a}}^{\underline{\boldsymbol{i}}}$, $\underline{x}^{\underline{\boldsymbol{i}}}$ and $\sum \underline{\boldsymbol{i}}$, respectively. We can write any element $f\left(\underline{x}\right) = f\left(x_1, \ldots, x_n\right) \in R[\underline{x}] \setminus \{0\}$ (sometimes we also write f simply) in a unique way as a sum of terms:

$$f\left(\underline{x}\right) = \sum_{\underline{\boldsymbol{i}} \in \Lambda} c_{\underline{\boldsymbol{i}}} \underline{x}^{\underline{\boldsymbol{i}}},$$

where Λ is the finite subset of $\left(\mathbb{Z}_{\geq 0}\right)^n$ and $c_{\underline{\boldsymbol{i}}} \in R \setminus \{0\}$ for $\underline{\boldsymbol{i}} \in \Lambda$. We then write $c_{\underline{\boldsymbol{i}}}(f) := c_{\underline{\boldsymbol{i}}}$ for $\underline{\boldsymbol{i}} \in \Lambda_f := \Lambda$. We call Λ_f the *support* of f. The total degree of f is denoted by w_f. For every element $\underline{\boldsymbol{a}} = (a_1, \ldots, a_n) \in R^n$ and invertible element $d \in R^\times$, we denote the element $(a_1/d, \ldots, a_n/d) \in R^n$ by $\underline{\boldsymbol{a}}/d$. Then we denote the value of $f\left(\underline{x}\right)$ at $\underline{\boldsymbol{a}}/d$ by $f\left(a_1/d, \ldots, a_n/d\right)$ or $f\left(\underline{\boldsymbol{a}}/d\right)$. In addition, if $R = \mathbb{Z}$ or $\mathbb{Q}$, then we use the following notation:

$$\begin{aligned} \Gamma_f &:= \{\left(\underline{\boldsymbol{i}}, b_{\underline{\boldsymbol{i}}}\right) \in \Lambda_f \times \mathbb{Z}_{>0} \,;\, 2^{b_{\underline{\boldsymbol{i}}}-1} \leq |c_{\underline{\boldsymbol{i}}}(f)| < 2^{b_{\underline{\boldsymbol{i}}}}\}, \\ H(f) &:= \max\{|c_{\underline{\boldsymbol{i}}}(f)| \,;\, \underline{\boldsymbol{i}} \in \Lambda_f\}. \end{aligned}$$

We call $H(f)$ the *height* of f. In addition, if for a polynomial $f \in \mathbb{Z}[\underline{x}]$, the support $\Lambda_f = \{\underline{\boldsymbol{i}}_1, \ldots, \underline{\boldsymbol{i}}_q\}$ is ordered by the order coming from the lexicographical order on

the monomials of f, then we denote by $\boldsymbol{f} = \left(c_{\boldsymbol{i}_1}(f), \ldots, c_{\boldsymbol{i}_q}(f)\right)$ the sequence of the ordered coefficients of f.

An m-dimensional lattice is defined as a discrete additive subgroup of an m-dimensional vector space over $\mathbb{R}$. It is well-known that for any lattice $\mathcal{L}$, there exist $\mathbb{R}$-linearly independent vectors generating $\mathcal{L}$ as a $\mathbb{Z}$-module. The rank of $\mathcal{L}$ is its rank as a $\mathbb{Z}$-module. For any lattice in $\mathbb{R}^m$ and its basis $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_r\}$, let $\boldsymbol{U}$ be an $r \times m$ matrix whose i-th row vector coincides with $\boldsymbol{b}_i$ for each i. Then we call $\boldsymbol{U}$ the *basis matrix* of the lattice. Let $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be the natural inner product for some $n \in \mathbb{Z}_{>0}$. For a vector $\boldsymbol{v} \in \mathbb{R}^n$, we denote the Euclidean norm of $\boldsymbol{v}$ by $\|\boldsymbol{v}\|$. We define the rounding function $\lfloor \cdot \rceil : \mathbb{R} \to \mathbb{Z}$ as $\lfloor c \rceil := \lfloor c + \frac{1}{2} \rfloor$ for any $c \in \mathbb{R}$. Let $\boldsymbol{M}$ be an $m \times n$ matrix over $\mathbb{Z}$ and $\varphi_{\boldsymbol{M}}$ the homomorphism as additive groups between $\mathbb{Z}^m \to \mathbb{Z}^n$ defined by $\boldsymbol{v} \mapsto \boldsymbol{v}\boldsymbol{M}$. Then the kernel of $\varphi_{\boldsymbol{M}}$ is a lattice in $\mathbb{R}^m$, and we call it the *kernel lattice* of $\boldsymbol{M}$.

2 The weighted LLL algorithm

In this section, we explain the weighted LLL algorithm, which is a key of our attack in Sect. 4, briefly. First, we define a weighted norm and a weighted lattice. They are useful for describing the weighted LLL algorithm.

Definition 1 Given a vector $\boldsymbol{w} = (w_1, \ldots, w_m) \in (\mathbb{R}_{>0})^m$, the *weighted norm* $\|\cdot\|_{\boldsymbol{w}} : \mathbb{R}^m \to \mathbb{R}$ for $\boldsymbol{w}$ is defined as follows:

$$\|\boldsymbol{a}\|_{\boldsymbol{w}} := \sqrt{(a_1 w_1)^2 + \cdots + (a_m w_m)^2} \quad , \text{where} \boldsymbol{a} = (a_1, \ldots, a_m) \in \mathbb{R}^m.$$

A *weighted lattice* for $\boldsymbol{w}$ in $\mathbb{R}^m$ is defined as a lattice endowed with the weighted norm for $\boldsymbol{w}$ (this means that we always mean the weighted norm on the weighted lattice when we consider a norm on the weighted lattice). Given a lattice $\mathcal{L} \subset \mathbb{R}^m$ and a vector $\boldsymbol{w} \in (\mathbb{R}_{>0})^m$, we denote $\mathcal{L}$ by $\mathcal{L}^{\boldsymbol{w}}$ whenever we endow $\mathcal{L}$ with the structure of a weighted lattice for $\boldsymbol{w}$.

For a lattice $\mathcal{L} \subset \mathbb{R}^m$ and a vector $\boldsymbol{w} = (w_1, \ldots, w_m) \in (\mathbb{R}_{>0})^m$, set a diagonal matrix $\boldsymbol{W}$ whose (i, i)-entry is w_i for $1 \leq i \leq m$. We consider the isomorphism $f_{\boldsymbol{W}} : \mathbb{R}^m \longrightarrow \mathbb{R}^m$ by $\boldsymbol{x} \mapsto \boldsymbol{x}\boldsymbol{W}$. Then, it is easy to show the equivalence of finding shortest lattice points, related with each other, in two lattices $\mathcal{L}^{\boldsymbol{w}}$ and $f_{\boldsymbol{W}}(\mathcal{L})$.

The weighted LLL algorithm for $\boldsymbol{w}$ is an algorithm to compute an LLL reduced basis (with respect to $\|\cdot\|_{\boldsymbol{w}}$) of $\mathcal{L}^{\boldsymbol{w}}$ (we call such a basis a weighted LLL reduced basis for $\boldsymbol{w}$ in this paper).

The most important lattice point in our attack is not necessarily shortest in a low-rank lattice, but only some of its entries are comparatively small. This property leads us to applying the weighted LLL algorithm to find such a lattice point by carefully controlling the entries of a weighted LLL reduced basis, see Sect. 4.3.

Remark 2 Controlling the entries of a basis output by the LLL algorithm is used in Coppersmith's method [9] and Faugére et al.'s method [14], see also Chapter 19 of

[16]. In their method, the scale of a lattice (or equivalently an inner product used in the LLL algorithm) is changed by heuristic ways. One can conduct such changes by changing a norm from the Euclidean norm to a weighted norm for some weight. In particular, our method for choosing a weighted norm is the same as the method in [14], see Step 1-2 of our algorithm in Sect. 4.2.

3 Brief review of DEC

In this section, we review DEC briefly, see Sect. 3 in [25] for details. As we mentioned in Sect. 1, DEC is constructed as a candidate of PQC and has the property, which is strongly desired in post-quantum cryptography, that sizes of public keys in DEC is small, e.g., about 1,200 bits with 128 bit security, see Remark 9. Note that sizes of public keys in cryptosystems [21,22,27], which are well-known to be efficient among the candidates of PQC, are about 10 times larger than 1,200 bits.

3.1 Definiton of polynomials of degree increasing type

Definition 3 Let $X\left(\underline{x}\right) \in \mathbb{Z}[\underline{x}]$ be a non-zero polynomial and define a map

$$\sigma : \mathbb{Z}^n \longrightarrow \mathbb{Z}_{\geq 0} \; ; \; \underline{\boldsymbol{i}} \mapsto \sum \underline{\boldsymbol{i}},$$

where we recall that $\sum \underline{\boldsymbol{i}} = \sum_{1 \leq k \leq n} i_k$ for $\underline{\boldsymbol{i}} = (i_1, \ldots, i_n)$. The polynomial X is *of degree increasing type* if $\sigma|_{\Lambda_X}$ is injective.

Remark 4 Let $X\left(\underline{x}\right)$ be a non-zero polynomial of $\mathbb{Z}[\underline{x}]$.

(1) From Definition 3, it is easy to see that $X\left(\underline{x}\right)$ is of degree increasing type if and only if the total degrees of the monomials of $X\left(\underline{x}\right)$ are different each other.
(2) Let X be a polynomial of degree increasing type. By the following order $\succ$, the support Λ_X becomes a totally ordered set: for two elements $(i_1, \ldots, i_n)$ and $(j_1, \ldots, j_n)$ in Λ_X, we have $(i_1, \ldots, i_n) \succ (j_1, \ldots, j_n)$ if and only if $i_1 + \cdots + i_n > j_1 + \cdots + j_n$.

Throughout this paper, whenever a polynomial X is of degree increasing type, we endow Λ_X with the total order given in Remark 4 (2).

Example 5 The polynomial $X(x, y, z) := 3x^3y^2z - 4x^2y^2 - xyz + 5yz + y + 11 \in \mathbb{Z}[x, y, z]$ is of degree increasing type.

Now, we describe DEC according to [25]. Note that Okumura did not suggest the security parameter because his purpose was to design the encryption scheme with 128 bit security. However, we here set the security parameter λ to analyse the complexity of our attack for each security level.

In accordance with [25], we regard the total degree of a public key polynomial as a parameter, which we denote by w_X. Note that the parameter w_X is taken to be an integer independent of the security parameter λ. In Remark 7 below, we will describe the reason why DEC has the two independent parameters λ and w_X.

3.2 Key generation process

Secret Key: A vector $\underline{\boldsymbol{a}} := (a_1, \ldots, a_n) \in \mathbb{Z}^n$.
Public Key:

(1) A positive integer d with $\gcd(a_i, d) = 1$ for all $1 \leq i \leq n$.
(2) A positive integer e with $\gcd(e, \varphi(d)) = 1$, where φ is the Euler function.
(3) A polynomial $X\left(\underline{x}\right) \in \mathbb{Z}[\underline{x}]$ of degree increasing type such that X is irreducible, $X\left(\underline{\boldsymbol{a}}/d\right) = 0$ and $\#\Lambda_X \leq w_X$, where Λ_X and w_X denote the support and the total degree of X, respectively.

Construction of $X\left(\underline{x}\right)$:

(1) Choose $\Lambda \subset \left(\mathbb{Z}_{\geq 0}\right)^n$ such that $3 \leq \#\{\sum \underline{\boldsymbol{i}}\ ;\ \underline{\boldsymbol{i}} \in \Lambda\} = \#\Lambda < \infty$ and $\underline{\mathbf{0}} \in \Lambda$, where $\underline{\mathbf{0}} := (0, \ldots, 0) \in \left(\mathbb{Z}_{\geq 0}\right)^n$.
(2) Let $\underline{\boldsymbol{k}}$ denote the maximal element of Λ (note that Λ is a totally ordered set in terms of the order given in Remark 4 (2)). Choose a random non-zero integer $c_{\underline{\boldsymbol{i}}}$ for each $\underline{\boldsymbol{i}} \in \Lambda \setminus \{\underline{\boldsymbol{k}}, \underline{\mathbf{0}}\}$. For a choice of $c_{\underline{\boldsymbol{i}}}$, see Remark 9 (2).
(3) Choose random integers $c_{\underline{\boldsymbol{k}}}$ and $c_{\underline{\mathbf{0}}}$ such that

$$c_{\underline{\boldsymbol{k}}}\underline{\boldsymbol{a}}^{\underline{\boldsymbol{k}}} + c_{\underline{\mathbf{0}}}d^w = -\sum_{\boldsymbol{k} \in \Lambda \setminus \{\underline{\boldsymbol{k}}, \underline{\mathbf{0}}\}} c_{\underline{\boldsymbol{i}}}\underline{\boldsymbol{a}}^{\underline{\boldsymbol{i}}}d^{w - \sum \underline{\boldsymbol{i}}}, \tag{1}$$

where $w := \max\{\sum \underline{\boldsymbol{i}}\ ;\ \underline{\boldsymbol{i}} \in \Lambda\}$.
(4) Set $\Lambda_X := \Lambda$ and $X\left(\underline{x}\right) := \sum_{\underline{\boldsymbol{i}} \in \Lambda_X} c_{\underline{\boldsymbol{i}}}\underline{x}^{\underline{\boldsymbol{i}}}$.

See Sect. 3.5 for a choice of a public key X and the sizes of the integers e, d and a_i's.

Remark 6 There exist integers $c_{\underline{\boldsymbol{k}}}$ and $c_{\underline{\mathbf{0}}}$ such that the equality (1) is satisfied because a_i and d are mutually prime for each $i \in \{1, \ldots, n\}$ from the assumption.

Remark 7 DEC has two parameters λ and w_X for the following reason: The public key of DEC is a Diophantine equation X of degree increasing type, and the secret key is its solution. Since there is no algorithm for solving Diophantine equations of degree increasing type, we set the security parameter, denoted by λ, which determines the security level against the key recovery attack by the brute force search (note that λ also determines the security level against some attacks on the one-way property of DEC, see [25]). On the other hand, w_x is an important parameter which complicates public diophantine equations and makes solving them difficult (by any method other than the brute force search), see also Remark 9.

3.3 Encryption process

Plaintext: A polynomial $m \in \mathbb{Z}[x_1, \ldots, x_n]$ such that

(a) $\Lambda_m = \Lambda_X$,
(b) $1 < c_{i_1, \ldots, i_n}(m) < d$ for all $(i_1, \ldots, i_n) \in \Lambda_m$,
(c) $\gcd\left(c_{i_1, \ldots, i_n}(m), d\right) = 1$ for all $(i_1, \ldots, i_n) \in \Lambda_m$.

Encryption Process:

(1) Choose a positive integer $N \in \mathbb{Z}_{>0}$ uniformly so that we have $Nd > 2^{\lambda} H(X)$. For a size of N, see Section 3.5 below.
(2) Construct $\widetilde{m}(\underline{x}) \in \mathbb{Z}[\underline{x}]$, called the twisted plaintext, by setting $\Lambda_{\widetilde{m}} := \Lambda_m$ and $c_{\underline{i}}(\widetilde{m}) := c_{\underline{i}}(m)^e \pmod{Nd}$, where $0 < c_{\underline{i}}(\widetilde{m}) < Nd$ for $\underline{i} \in \Lambda_{\widetilde{m}}$.
(3) Choose $f(\underline{x}) \in \mathbb{Z}[\underline{x}]$ uniformly at random such that
   (a) $\Lambda_f = \Lambda_X$,
   (b) $H(\widetilde{m}) < c_{\underline{k}}(f) < Nd$ and $\gcd(c_{\underline{k}}(f), d) = 1$, where $\underline{k}$ denotes the maximal element of Λ_f.
(4) Choose $s_j(\underline{x}), r_j(\underline{x}) \in \mathbb{Z}[\underline{x}]$ uniformly at random so that we have $\Gamma_{s_j} = \Gamma_X$ and $\Gamma_{r_j} = \Gamma_f$ for $1 \le j \le 3$.
(5) Put $F_j(\underline{x}) := \widetilde{m}(\underline{x}) + s_j(\underline{x}) f(\underline{x}) + r_j(\underline{x}) X(\underline{x})$ for $1 \le j \le 3$. Send (F_1, F_2, F_3, N) as a ciphertext.

3.4 Decryption process

Decryption Process:

(1) By substituting $\underline{a}/d$, a zero of $X(\underline{x})$, into $F_j(\underline{x})$, we obtain

$$h_j := F_j(\underline{a}/d) = \widetilde{m}(\underline{a}/d) + s_j(\underline{a}/d) f(\underline{a}/d) \text{ for } 1 \le j \le 3.$$

Compute

$$H_1 := (h_1 - h_2) d^{2w_X} = (s_1(\underline{a}/d) - s_2(\underline{a}/d)) f(\underline{a}/d) d^{2w_X},$$
$$H_2 := (h_1 - h_3) d^{2w_X} = (s_1(\underline{a}/d) - s_3(\underline{a}/d)) f(\underline{a}/d) d^{2w_X}.$$

(2) Compute $g := \gcd(H_1, H_2)$. If $\gcd(g, d) > 1$, then let d' be the smallest factor of g satisfying $\gcd(d, g/d') = 1$ and replace g by g/d'.
(3) Compute $H := h_1 d^{2w_X} \pmod{g}$ and $\mu := H d^{-w_X} \pmod{g}$.
(4) Obtain the plaintext polynomial $m(\underline{x})$ from μ or $\mu - g$ by using an algorithm described in Sects. 3.4 and 3.5 of [25].

Remark 8 In the algorithm in Sects. 3.4 and 3.5 of [25], we need to compute $\varphi(d)$ efficiently. From this, we should choose a prime number as d.

3.5 Parameter size

In Sect. 5 of [25], sizes of public/secret keys and ciphertexts are estimated so that DEC can be expected to have 128 bit security under some assumptions. In the following, we give their sizes under the same assumptions as [25] to analyse the complexity of our attack.

(1) The sizes of $\underline{\boldsymbol{a}}$, d, e and N:

$$2^{\frac{\lambda}{2}} \leq d < 2^{\frac{\lambda}{2}+1},\ (\lambda+1)+\left(\frac{\lambda}{2}+1\right)w_X \leq e < 2\left((\lambda+1)+\left(\frac{\lambda}{2}+1\right)w_X\right),$$
$$\frac{2^{\left\lceil\frac{\lambda}{n-1}\right\rceil}}{\varphi(d)}d \leq |a_i| < \frac{2^{\left\lceil\frac{\lambda}{n-1}\right\rceil+1}}{\varphi(d)}d\ (1 \leq i \leq n),$$
$$2^{\lambda+\left(\frac{\lambda}{2}+1\right)(w_X-1)} \leq N < 2^{\lambda+1+\left(\frac{\lambda}{2}+1\right)(w_X-1)}.$$

We assume that $|c_{\underline{\boldsymbol{i}}}(X)| < 2^b$ for any $\underline{\boldsymbol{i}} \in \Lambda_X \setminus \{\underline{\boldsymbol{k}}, \underline{\boldsymbol{0}}\}$, where $\underline{\boldsymbol{k}}$ denotes the maximal element of Λ_X, see Sect. 5 of [25].

(2) The size of a secret key is at most

$$\left(\left\lceil\frac{\lambda}{n-1}\right\rceil+1\right)n+\lceil\log_2 d-\log_2\varphi(d)\rceil$$

bits.

(3) The size of a public key is at most

$$\left(\left\lceil\frac{\lambda}{n-1}\right\rceil+\left(\frac{\lambda}{2}+2+b\right)+\lceil\log_2 d-\log_2\varphi(d)\rceil\right)w_X+(\lambda+1)+\lceil\log_2 e\rceil$$

bits.

(4) The size of a ciphertext is at most

$$\frac{3}{2}\left(w_X^2+w_X\right)\left(\lambda+1+(\lambda+2)w_X+\lceil\log_2 w_X\rceil\right)+\lambda+1+\left(\frac{\lambda}{2}+1\right)(w_X-1)$$

bits. Note that the size of each coefficient of F_i is at most

$$\lambda+1+(\lambda+2)w_X+\lceil\log_2 w_X\rceil$$

bits for $i = 1, 2$, and 3.

Remark 9 (1) In Sect. 4.5 of [25], it is pointed out that we should use a polynomial X satisfying $w_X \geq 5$, $n \geq 3$ and some conditions as a public key in order to avoid finding rational solutions to $X = 0$. However, polynomials of degree increasing type are in a special class of polynomials, and finding rational zeros of such polynomials may be easier than finding those of general polynomials. Moreover, although finding rational zeros of polynomials of higher degree seems to be difficult in general, we should consider sizes of public keys and ciphertexts. Thus we recommend to use X of degree 10 as a public key.

(2) In Sect. 5 of [25], it is pointed out that for a public key X and $\underline{\boldsymbol{i}} \in \Lambda_X \setminus \{\underline{\boldsymbol{k}}, \underline{\boldsymbol{0}}\}$, we may choose $c_{\underline{\boldsymbol{i}}}(X) \leq 2^{10}$, where $\underline{\boldsymbol{k}}$ denotes the maximal element of Λ_X. However, since solving Diophantine equations of degree increasing type may be easier than

solving more general Diophantine equations as we mentioned above, we should also consider using larger $c_{\underline{i}}(X)$ for $\underline{i} \in \Lambda_X \setminus \{\underline{k}, \underline{0}\}$ to deal with a wide class of polynomials of degree increasing type. In our experiments of Sect. 6, we choose $c_{\underline{i}}(X)$ so that the sizes of $\left|c_{\underline{i}}(X)\right|$ are b bits for $b = 10, 50$ and 100.

(3) When $\lambda = 128, w_X = \#\Lambda_X$ and $b = 10$, we generated 100 public keys X randomly and measured their sizes. As a result, their average size is about 1,200 bits. This size (1,200 bits) is about 10 times smaller than sizes of public keys in cryptosystems [21,22,27], which are well-known to be efficient among the candidates of PQC.

3.6 Toy example of DEC

In the following, we give a toy example of DEC in the case of $n = 2$.

Secret Key: $\underline{a} = (a, b) = (47, 49) \in \mathbb{Z}^2$.
Public Key: $(d, e, X) = (5, 17, 125x^3 + 675y - 110438)$.
$(\Lambda_X = \{(3, 0), (0, 1), (0, 0)\}, \underline{k} = (3, 0), H(X) = 110438.)$
Plaintext: $m\left(\underline{x}\right) = m(x, y) = 3x^3 + 3y + 2$.
Objects for Encryption:

(1) $N = 353408$ $(Nd = 1767040)$.
(2) $\widetilde{m}\left(\underline{x}\right) = \widetilde{m}(x, y) = 146243x^3 + 146243y + 131072$ $(H(\widetilde{m}) = 146243)$.
(3) $f\left(\underline{x}\right) = f(x, y) = 949843x^3 + 1324952y + 1109775$.
$(c_{\underline{k}}(f) = 949843, H(\widetilde{m}) = 146243 < c_{\underline{k}}(f) = 949843 < 1767040 = Nd.)$
(4) s_j and r_j:

$$s_1 = 115x^3 + 924y + 126337, \quad s_2 = 82x^3 + 962y + 89939,$$
$$s_3 = 67x^3 + 977y + 121816, \quad r_1 = 691019x^3 + 1363650y + 1329029,$$
$$r_2 = 852655x^3 + 1584164y + 2007688,$$
$$r_3 = 940020x^3 + 2016302y + 1144882.$$

(5) *Cipher Polynomials*: $F_j := \widetilde{m} + s_j f + r_j X$.

$$\begin{aligned} F_1 &= 195609320x^6 + 1666918487x^3y + 43979457762x^3 + 2144719398y^2 \\ &\quad +18714355042y - 6569529455, \\ F_2 &= 184469001x^6 + 1795957655x^3y - 8395474520x^3 + 2343914524y^2 \\ &\quad -53364106711y - 121912862547, \\ F_3 &= 181141981x^6 + 1903319645x^3y + 12109757546x^3 + 2655481954y^2 \\ &\quad -59418815676y + 8750004156. \end{aligned}$$

4 Weighted LLL-based polynomial time-attack for DEC

We give in this section our attack algorithm against DEC, based on the weighted LLL. We use the following notation described in Notation of Sect. 1: for a polynomial $h =$

$\sum_{\underline{\boldsymbol{i}} \in \Lambda_h} c_{\underline{\boldsymbol{i}}}(h) x^{\underline{\boldsymbol{i}}} \in \mathbb{Z}[\underline{x}]$, let $\boldsymbol{h} := \left(c_{\underline{\boldsymbol{i}}_1}(h), \ldots, c_{\underline{\boldsymbol{i}}_{\#\Lambda_h}}(h)\right)$, where $\Lambda_h = \{\underline{\boldsymbol{i}}_1, \ldots, \underline{\boldsymbol{i}}_{\#\Lambda_h}\}$ is the support of h. Note that Λ_h is an ordered set (see Notation in Sect. 1).

Let $(d, e, X(\underline{x})) \in \mathbb{Z}^2 \times (\mathbb{Z}[\underline{x}])$ and $(F_1(\underline{x}), F_2(\underline{x}), F_3(\underline{x}), N) \in (\mathbb{Z}[\underline{x}])^3 \times \mathbb{Z}$ be a public key and a ciphertext, as described in Sects. 3.2 and 3.3. Let $m(\underline{x}) \in \mathbb{Z}[\underline{x}]$ be a plaintext. Each cipher polynomial is of the form $F_j(\underline{x}) = \widetilde{m}(\underline{x}) + s_j(\underline{x}) f(\underline{x}) + r_j(\underline{x}) X(\underline{x})$ for the twisted plaintext $\widetilde{m}(\underline{x})$ and some random polynomials $f(\underline{x})$, $s_j(\underline{x})$ and $r_j(\underline{x})$. For the choice of $f(\underline{x})$, $s_j(\underline{x})$ and $r_j(\underline{x})$, see Sect. 3.3 for details. We write $\Lambda_X = \{\underline{\boldsymbol{i}}_1, \ldots, \underline{\boldsymbol{i}}_q\}$ with $\underline{\boldsymbol{i}}_1 \succ \cdots \succ \underline{\boldsymbol{i}}_q$, where the total order $\succ$ on Λ_X is given in Remark 4 (2). Recall that the supports of m, $\widetilde{m}$, s_j, r_j and f $(1 \leq j \leq 3)$ are the same as Λ_X, which allows attackers to suppose $\Lambda_{F_1} = \Lambda_{F_2} = \Lambda_{F_3}$. Let $\underline{\boldsymbol{k}}$ denote the maximal element of Λ_X. To simplify the notation, put $q := \#\Lambda_X$ throughout this section. For recovering m, it suffices to get the correct $\widetilde{m}$.

4.1 Idea of our attack

Before we give an algorithm of our attack, we describe the idea of our attack. Recall from Sect. 3 that in DEC, we use the cipher polynomials of the form

$$F_j := \widetilde{m} + s_j f + r_j X \quad \text{for } j = 1, 2 \text{ and } 3.$$

We reduce recovering $\widetilde{m}$ to finding special solutions to certain linear systems derived from X and (F_1, F_2, F_3, N), the public key and the ciphertext, by linearization techniques described below.

We have the following equalities for $j = 1$ and 2 from the way to construct the cipher polynomials:

$$F_j - F_{j+1} = (s_j - s_{j+1}) f + (r_j - r_{j+1}) X.$$

Since the cipher polynomials $F_1(\underline{x})$, $F_2(\underline{x})$, $F_3(\underline{x})$ and the public key $X(\underline{x})$ are known, we may obtain $f(\underline{x})$ if we determine $s_1(\underline{x}) - s_2(\underline{x})$ and $s_2(\underline{x}) - s_3(\underline{x})$. We set

$$\begin{aligned} s'_j &:= s_j - s_{j+1}, \quad r'_j := r_j - r_{j+1}, \\ F'_j &:= F_j - F_{j+1} = s'_j f + r'_j X \quad \text{for } j = 1 \text{ and } 2, \\ g &:= s'_2 r'_1 - s'_1 r'_2. \end{aligned}$$

We then have the following equalities:

$$F'_1(\underline{x}) = s'_1(\underline{x}) f(\underline{x}) + r'_1(\underline{x}) X(\underline{x}), \tag{2}$$

$$F'_2(\underline{x}) = s'_2(\underline{x}) f(\underline{x}) + r'_2(\underline{x}) X(\underline{x}), \tag{3}$$

$$g(\underline{x}) X(\underline{x}) = s'_2(\underline{x}) F'_1(\underline{x}) - s'_1(\underline{x}) F'_2(\underline{x}). \tag{4}$$

4.1.1 First step: determination of s'_j for $j = 1$ and 2

Here, we describe how to determine $\boldsymbol{s}'_j$ for $j = 1$ and 2. (As we mentioned in Sect. 1, the vectors $\boldsymbol{s}'_j$ ($j = 1$ and 2) are the most important target vectors). In the equality (4), we regard the coefficients of $s'_j(\underline{x})$ and $g(\underline{x})$ as indeterminates. We then obtain the linear system $\boldsymbol{u}\boldsymbol{A} = \boldsymbol{0}$, where $\boldsymbol{A}$ is the $\left(\left(2q + \#\Lambda_{X^2}\right) \times \#\Lambda_{X^3}\right)$ coefficient matrix of the linear system. We denote by $\mathcal{L}'_1$ the kernel lattice of $\boldsymbol{A}$, where the kernel lattice of $\boldsymbol{A}$ is defined as the nullspace of $\boldsymbol{A}$, see Notation in Sect. 1. Let $\mathcal{L}_1$ be the lattice spanned by the vectors consisting of the 1-$(2q)$th entries of the elements in $\mathcal{L}'_1$. Experimentally, the rank of $\mathcal{L}_1$ is equal to 3 in many cases, see Remark 27 in Sect. 6. Thus, we assume the following condition:

Assumption 10 The rank of $\mathcal{L}_1$ is equal to 3.

Moreover, as we will see in Sect. 4.3, the correct $\left(\boldsymbol{s}'_1, \boldsymbol{s}'_2\right)$ has the property described in Sects. 1 and 2 so that the usual LLL reduction does not work well to find $\left(\boldsymbol{s}'_1, \boldsymbol{s}'_2\right)$. Note that this is true in many cases because of the construction of X (cf. Sect. 3.2). Thus, we use the weighted LLL reduction for a weight $\boldsymbol{w}$ described below. Put $\boldsymbol{w}' = \left(w'_1, \ldots, w'_q\right)$ as follows:

$$w'_j := 2^{\left\lfloor \log_2\left(\frac{H(X)}{c_{\underline{i}_j}}\right)\right\rfloor},$$

where $\boldsymbol{X} := (c_{\underline{i}_1}(X), \ldots, c_{\underline{i}_q}(X))$ denotes the vector of the coefficients of $X(\underline{x})$. We set $\boldsymbol{w} := \left(w'_1, \ldots, w'_q, w'_1, \ldots, w'_q\right)$. Assume the following condition.

Assumption 11 The $\left(\boldsymbol{s}'_1, \boldsymbol{s}'_2\right)$ is a shortest vector in $\mathcal{L}_1^{\boldsymbol{w}}$.

Let $f_{\boldsymbol{W}}$ be the isomorphism described in Sect. 2 from $\mathbb{R}^{2q}$ to $\mathbb{R}^{2q}$ as $\mathbb{R}$-vector spaces. From Assumption 10, the rank of $f_{\boldsymbol{W}}(\mathcal{L}_1)$ is equal to 3. This means that we can expect the weighted LLL reduction for the weight $\boldsymbol{w}$ to output a shortest vector in $\mathcal{L}_1^{\boldsymbol{w}}$ with high probability. Thus it is expected to find the correct $\left(\boldsymbol{s}'_1, \boldsymbol{s}'_2\right)$ via the weighted LLL reduction for the weight $\boldsymbol{w}$, see Sect. 2 and Assumption 11.

Remark 12 As we will see in Sect. 4.3, one may fail in determining $(\boldsymbol{s}'_1, \boldsymbol{s}'_2)$ even if one adopts the LLL reduction in terms of the p-norm ($1 \leq p \leq \infty$) as a lattice reduction for $\mathcal{L}_1$. Thus, the above assumptions and applying the weighted LLL reduction to $\mathcal{L}_1$ are crucial for our attack.

4.1.2 Second step: obtaining a candidate of f

Here, we describe how to determine a candidate of f. We substitute $s'_1(\underline{x})$ and $s'_2(\underline{x})$ obtained in Step 1 into (2) and (3). In a similar way to Step 1, by regarding the coefficients of $f(\underline{x})$ and $r'_j(\underline{x})$ for $j = 1$ and 2 as indeterminates, we have the linear system. We then fix $f'(\underline{x})$ such that (2) and (3) hold and that $f'(\underline{x})$ is close

to the correct $f(\underline{x})$, i.e., the absolute values of all coefficients of the polynomial $f'(\underline{x}) - f(\underline{x})$ are small. Note that $f'(\underline{x})$ does not necessarily coincide with the correct $f(\underline{x})$ to recover $\widetilde{m}$ (cf. Remark 18 and Steps 3-3 and 3-4 in Sect. 4.2).

Remark 13 In Step 2, any solution (f', r_1'') to the linear system can be written as $f' = f + aX$ and $r_1'' = r_1' - as_1'$, respectively ($a \in \mathbb{Z}$) if $\gcd(X, s_1') = 1$ and if the solution in Step 1 is the correct (s_1', s_2'). In fact, by putting $p := f' - f$ and $q := r_1'' - r_1'$, we have

$$\begin{aligned} F_1' &= s_1' f' + r_1'' X \\ &= s_1'(f + p) + (r_1' + q) X \\ &= (s_1' f + r_1' X) + (s_1' p + qX) \\ &= F_1' + (s_1' p + qX). \end{aligned}$$

It follows that $s_1' p = -qX$. Thus if $\gcd(X, s_1') = 1$, there exists an integer $a \in \mathbb{Z}$ such that $p = aX$ and $q = -as_1'$ since $\deg p \leq \deg X$ and $\deg q \leq \deg s_1'$. This fact implies that the rank of the kernel lattice in Step 2 is equal to 1 if $\gcd(X, s_1') = 1$. If the solution obtained in Step 1 is $(-s_1', -s_2')$, then (f', r_1'') can be written as $f' = -f + aX$ and $r_1'' = r_1' + as_1'$, respectively ($a \in \mathbb{Z}$) by the same argument. Note that since X is irreducible from the construction of X in Sect. 3.2, we have $\gcd(X, s_1') = 1$ with high probability.

4.1.3 Third step: recovery of $\widetilde{m}$

Here, we describe how to recover $\widetilde{m}$. It is sufficient for recovering $\widetilde{m}(\underline{x})$ to find s_1, see Remark 18 and Steps 3-3 and 3-4 in Sect. 4.2. From the form of the ciphertext (see Sect. 3.3), consider the following equality:

$$F_1 = \widetilde{m} + s_1 f' + r_1 X, \tag{5}$$

where $f'(\underline{x})$ is the polynomial obtained in Step 2 and other polynomials $\widetilde{m}(\underline{x})$, $s_1(\underline{x})$ and $r_1(\underline{x})$ are unknown. Note that if we have the correct solution in Step 1 and $\gcd(X, s_1') = 1$, then there exists a unique polynomial $r(\underline{x})$ such that the correct $\widetilde{m}(\underline{x})$, $s_1(\underline{x})$ and $f'(\underline{x})$ (not necessarily $f(\underline{x})$) satisfy the equality $F_1 = \widetilde{m} + s_1 f' + rX$, see Remark 18. In a similar way to Steps 1 and 2, by regarding the coefficients of $\widetilde{m}(\underline{x})$, $s_1(\underline{x})$ and $r_1(\underline{x})$ as indeterminates, we have the linear system $\boldsymbol{w}\boldsymbol{C} = \boldsymbol{c}$, where $\boldsymbol{C}$ is the $(3q \times \#\Lambda_{X^2})$ coefficient matrix of the linear system and $\boldsymbol{c} \in \mathbb{Z}^{\#\Lambda_{X^2}}$. We denote by $\mathcal{L}_3$ the kernel lattice of $\boldsymbol{C}$. The rank of $\mathcal{L}_3$ is equal to 3 with high probability, see Remark 27. From this, we assume the following:

Assumption 14 The rank of $\mathcal{L}_3$ is equal to 3.

Let $\boldsymbol{w}_0$ be one solution to $\boldsymbol{w}\boldsymbol{C} = \boldsymbol{c}$ and $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3\}$ a basis of $\mathcal{L}_3$. Note that every integral solution to the system is represented as $\boldsymbol{w}_0 + a_1\boldsymbol{w}_1 + a_2\boldsymbol{w}_2 + a_3\boldsymbol{w}_3$ ($a_i \in \mathbb{Z}$, $i = 1, 2$ and 3). The 1-$\#\Lambda_X$-th entries of $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$ and $\boldsymbol{w}_3$ correspond to the coefficients of $\widetilde{m}$. As we will see in Remark 17, the system $\boldsymbol{w}\boldsymbol{C} = \boldsymbol{0}$ has a solution

$\boldsymbol{w}'$ whose 1-$\#\Lambda_X$-th entries equal zero. We choose such a solution as $\boldsymbol{w}_3$. Assume the following condition:

Assumption 15 The entries in $\boldsymbol{s}_1$ coincide with the $(\#\Lambda_X + 1)$-$2\#\Lambda_X$-th entries in $\boldsymbol{w}_0 + \boldsymbol{w}_3 - \boldsymbol{z}$, where $\boldsymbol{z}$ is a closest lattice point in $\mathcal{L}_3' := \langle \boldsymbol{w}_1, \boldsymbol{w}_2 \rangle_{\mathbb{Z}}$ to $\boldsymbol{w}_0 + \boldsymbol{w}_3$. In other words, $\boldsymbol{s}_1$ is embedded in $\boldsymbol{w}_0 + \boldsymbol{w}_3 - \boldsymbol{z}$ as its $(\#\Lambda_X + 1)$-$2\#\Lambda_X$-th entries.

The lattice $\mathcal{L}_3'$ has rank 2, and thus we can expect to find $\boldsymbol{s}_1$ in polynomial time by the Babai nearest plane algorithm [5] for solving CVP with sufficiently high probability under Assumption 15.

Remark 16 The reason why we assume Assumption 15 is the following: From the choice of $\boldsymbol{s}_1$, the absolute values of the entries of $\boldsymbol{s}_1$ are sufficiently smaller than those of $\widetilde{\boldsymbol{m}}$ and $\boldsymbol{r}_1$. Thus we can expect that the value of $\|\boldsymbol{w}_0 + \boldsymbol{w}_3 - (a_1\boldsymbol{w}_1 + a_2\boldsymbol{w}_2)\|$ is sufficiently small if certain entries of the vector $\boldsymbol{w}_0 + \boldsymbol{w}_3 - (a_1\boldsymbol{w}_1 + a_2\boldsymbol{w}_2)$ coincide with those of $\boldsymbol{s}_1$.

Remark 17 In Step 3, the linear system $\boldsymbol{wC} = \boldsymbol{0}$ has a solution $\boldsymbol{w}'$ whose 1-$\#\Lambda_X$-th entries equal zero. Let $(\boldsymbol{m}', \boldsymbol{s}', \boldsymbol{r}')$ be one solution to $\boldsymbol{wC} = \boldsymbol{c}$, i.e., $F_1 = m' + s'f' + r'X$. The vector $(\boldsymbol{m}', \boldsymbol{s}', \boldsymbol{r}') + (\boldsymbol{0}, \boldsymbol{X}, -\boldsymbol{f}')$ is also a solution to $\boldsymbol{wC} = \boldsymbol{c}$. In fact, we have

$$(m' + 0) + (s' + X)\, f' + (r' - f')\, X = (m' + s'f' + r'X) + Xf' - f'X = F_1.$$

Thus $(\boldsymbol{0}, \boldsymbol{X}, -\boldsymbol{f}')$ is an element of $\mathcal{L}_3$.

Remark 18 If we succeed in finding the correct s_1 in Step 3 and $\gcd(X, s_1') = 1$, there exists r satisfying the equality $F_1 - s_1 f' = \widetilde{m} + rX$. In fact, f' obtained in Step 2 can be written as $f' = f + aX$ or $f' = -f + aX$ $(a \in \mathbb{Z})$ from Remark 13. We may assume that $f' = f + aX$. Then we have

$$\begin{aligned} F_1 - \widetilde{m} - s_1 f' &= s_1 f + r_1 X - s_1 f' \\ &= s_1 \left(f' - aX\right) + r_1 X - s_1 f' \\ &= (r_1 - a s_1)\, X. \end{aligned}$$

Thus we have $F_1 - s_1 f' = \widetilde{m} + rX$ by putting $r := r_1 - a s_1$.

4.2 Algorithm of our attack

Based on the idea in Sect. 4.1, we write down our attack algorithm against DEC in what follows. Let $(d, e, X(\underline{x})) \in \mathbb{Z}^2 \times \mathbb{Z}[\underline{x}]$ and $(F_1(\underline{x}), F_2(\underline{x}), F_3(\underline{x}), N) \in (\mathbb{Z}[\underline{x}])^3 \times \mathbb{Z}$ be a public key and a ciphertext, as described in Sects 3.2 and 3.3. Let $m(\underline{x}) \in \mathbb{Z}[\underline{x}]$ be a plaintext. Each cipher polynomial is of the form $F_j(\underline{x}) = \widetilde{m}(\underline{x}) + s_j(\underline{x}) f(\underline{x}) + r_j(\underline{x}) X(\underline{x})$ for the twisted plaintext $\widetilde{m}(\underline{x})$ and some random polynomials $f(\underline{x})$, $s_j(\underline{x})$ and $r_j(\underline{x})$. We also recall that Λ_X and w_X denote the support of X and the total degree of X, respectively, see Notation in Sect. 1. Let $\underline{\boldsymbol{k}}$ be the maximal element of Λ_X, see Remark 4 (2) for the ordering.

Attack Algorithm

Input: $\left(d, e, X(\underline{x})\right)$ and $\left(F_1(\underline{x}), F_2(\underline{x}), F_3(\underline{x}), N\right)$, a public key and a ciphertext.
Output: $\widetilde{m}\left(\underline{x}\right)$, a twisted plaintext.

Step 1: Determination of $s'_j := s_j - s_{j+1}$ for $j = 1$ and 2

Step 1-1: Put $F'_j := F_j - F_{j+1}$, $r'_j := r_j - r_{j+1}$ $(1 \leq j \leq 2)$ and put $g := s'_2 r'_1 - s'_1 r'_2$. Compute a basis of the kernel lattice of $\boldsymbol{A}$, i.e., solve $\boldsymbol{u}\boldsymbol{A} = \boldsymbol{0}$. This system is derived from unknown coefficients in

$$s'_2 F'_1 - s'_1 F'_2 = gX, \tag{6}$$

where $\boldsymbol{A}$ is the $\left(2\#\Lambda_X + \#\Lambda_{X^2}\right) \times \#\Lambda_{X^3}$ coefficient matrix of the linear system obtained from the Eq. (6). Let $\{\boldsymbol{u}'_1, \boldsymbol{u}'_2, \boldsymbol{u}'_3\}$ be the set of basis vectors for the kernel lattice.
Step 1-2: We denote by $\boldsymbol{u}_i$ the vector embedded in $\boldsymbol{u}'_i$ as its 1-$(2\#\Lambda_X)$-th entries for $i = 1, 2$ and 3. Execute the weighted LLL reduction for the weight described in Sect. 4.1 to the lattice $\mathcal{L}_1 := \langle \boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3 \rangle$, and then get $(\boldsymbol{s}'_1, \boldsymbol{s}'_2)$.

Step 2: Obtaining a candidate of f

Step 2-1: Compute a solution to $\boldsymbol{v}\boldsymbol{B} = \boldsymbol{b}$. This system is derived from unknown coefficients in

$$F'_1 = s'_1 f + r'_1 X, \quad F'_2 = s'_2 f + r'_2 X, \tag{7}$$

where $\boldsymbol{B}$ is the $\left(3\#\Lambda_X \times \#\Lambda_{X^2}\right)$ coefficient matrix obtained from the Eq. (7). Let $\boldsymbol{v}_0$ be a solution, and let $\{\boldsymbol{v}_1\}$ be a basis of the kernel lattice $\mathcal{L}_2$ of $\boldsymbol{B}$. If $\gcd(X, s'_1) = 1$ in $\mathbb{Z}[\underline{x}]$, then the lattice $\mathcal{L}_2$ always has rank 1, see Remark 13.
Step 2-2: Compute $\boldsymbol{v}'_0 := \boldsymbol{v}_0 - \lfloor \langle \boldsymbol{v}_0, \boldsymbol{v}_1 \rangle / \langle \boldsymbol{v}_1, \boldsymbol{v}_1 \rangle \rceil \boldsymbol{v}_1$, another solution to $\boldsymbol{v}\boldsymbol{B} = \boldsymbol{b}$. Let $\boldsymbol{v}''_0$ be the vector embedded in $\boldsymbol{v}'_0$ as its 1-$(\#\Lambda_X)$-th entries. Let $f'\left(\underline{x}\right) \in \mathbb{Z}[\underline{x}]$ be a polynomial with $\boldsymbol{f}' = \boldsymbol{v}''_0$. Experimentally $\boldsymbol{v}'_0$ gives in many cases a polynomial closer to f than $\boldsymbol{v}_0$, see Step 2 in Sect. 4.3.

Step 3: Recovery of $\widetilde{m}$

Step 3-1: Compute a solution to $\boldsymbol{w}\boldsymbol{C} = \boldsymbol{c}$ and a basis of the kernel lattice of $\boldsymbol{C}$. This system is derived from unknown coefficients in

$$F_1 = \widetilde{m} + s_1 f' + r_1 X, \tag{8}$$

where $\boldsymbol{C}$ is the $(3\#\Lambda_X \times \#\Lambda_{X^2})$ coefficient matrix oftained from the Eq. (8) and f' is the polynomial obtained in Step 2-2. Let $\boldsymbol{w}_0$ be a solution and $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3\}$ a basis of the kernel lattice, denoted by $\mathcal{L}_3$.
Step 3-2: Apply the Babai nearest plane algorithm to compute a closest lattice point $\boldsymbol{z}$ in $\mathcal{L}'_3 := \langle \boldsymbol{w}_1, \boldsymbol{w}_2 \rangle_{\mathbb{Z}}$ to $\boldsymbol{w}_0 + \boldsymbol{w}_3$. Let $\boldsymbol{s}_1$ be the vector embedded in $\boldsymbol{w}_0 + \boldsymbol{w}_3 - \boldsymbol{z}$ as its $(\#\Lambda_X + 1)$-$2\#\Lambda_X$-th entries.

Step 3-3: Compute a solution to $\boldsymbol{xH} = \boldsymbol{h}$, the linear system derived from unknown coefficients in

$$F_1 - \widetilde{m} - s_1 f' = rX, \tag{9}$$

where $\boldsymbol{H}$ is the $\left(2\#\Lambda_X \times \#\Lambda_{X^2}\right)$ coefficient matrix obtained from the Eq. (9) and the coefficients of $\widetilde{m}$ and r are indeterminates. Let $\boldsymbol{x}$ be a solution to $\boldsymbol{xH} = \boldsymbol{h}$. Let $\boldsymbol{r}'$ be the vector consisting of the entries corresponding to r of $\boldsymbol{x}$. Then we obtain a polynomial r' whose coefficients coincide with those of r except the constant part, i.e., $r = r' + t$ for some $t \in \mathbb{Z}$.
Step 3-4: Compute

$$\begin{aligned}
e' &:= e^{-1} \pmod{\varphi(d)},\\
H_1 &:= F_1 - s_1 f' - r'X,\\
\mu &:= c_{\underline{\boldsymbol{k}}}(H_1),\\
c_{\underline{\boldsymbol{k}}}(m') &:= \mu^{e'} \pmod{d} \quad \left(0 < c_{\underline{\boldsymbol{k}}}(m') < d\right),\\
c_{\underline{\boldsymbol{k}}}(\widetilde{m}) &:= \left(c_{\underline{\boldsymbol{k}}}(m')\right)^e \pmod{Nd} \quad \left(0 < c_{\underline{\boldsymbol{k}}}(\widetilde{m}) < Nd\right),\\
t &:= \left(\mu - c_{\underline{\boldsymbol{k}}}(\widetilde{m})\right)/c_{\underline{\boldsymbol{k}}}(X),\\
\widetilde{m} &:= F_1 - s_1 f' - \left(r' + t\right)X.
\end{aligned}$$

Output $\widetilde{m}\left(\underline{x}\right)$.

Remark 19 One may consider that applying the Babai nearest plane algorithm in terms of a weighted norm, or searching a desired vector $\boldsymbol{s}_1$ by adding some elements in $\mathcal{L}_3'$ are effective. However, the one-way property of DEC can be broken with sufficiently high probability without such operations. We will see the details in Sect. 6. Hence in our attack let us omit these procedures.

Remark 20 In Step 3-4 of the above algorithm, we use the fact that $c_{\underline{\boldsymbol{k}}}(X)$ is divisible by d to compute an integer t, see (1) for the divisibility of $c_{\underline{\boldsymbol{k}}}(X)$.

4.3 Cryptanalysis of toy example

We break the one-way property of the instance in Sect. 3.6 of DEC. We use the same notations as in Sect. 3.6. In this case, we have $\Lambda_g = \Lambda_{X^2} = \{(6,0), (3,1), (3,0), (0,2), (0,1), (0,0)\}$.

4.3.1 First step: determination of $s_j' = s_j - s_{j+1}$

Here, we determine $s_j' = s_j - s_{j+1}$ for $j = 1$ and 2. Compute

$$\begin{aligned}
F_1' := F_1 - F_2 = {} & 11140319x^6 - 129039168x^3y + 52374932282x^3 - 199195126y^2\\
& + 72078461753y + 115343333092,\\
F_2' := F_2 - F_3 = {} & 3327020x^6 - 107361990x^3y - 20505232066x^3 - 311567430y^2\\
& + 6054708965y - 130662866703.
\end{aligned}$$

We put

$$s'_j := c_1^{(j)}x^3 + c_2^{(j)}y + c_3^{(j)} \quad \text{for } j = 1 \text{ and } 2,$$
$$g := c_1^{(g)}x^6 + c_2^{(g)}x^3y + c_3^{(g)}x^3 + c_4^{(g)}y^2 + c_5^{(g)}y + c_6^{(g)},$$

where $c_i^{(j)}$'s and $c_i^{(g)}$'s are indeterminates. By comparing the coefficient of $\underline{x}^{\underline{i}}$ for each $\underline{i} \in \Lambda_{X^3}$ in the Eq. (4), we have the linear system $\boldsymbol{u}\boldsymbol{A}' = \boldsymbol{0}$, where $\boldsymbol{A}'$ is a (9×9) matrix.

The rank of the kernel lattice $\mathcal{L}'_1$ of $\boldsymbol{A}'$ is equal to 3. Compute a basis $\{\boldsymbol{u}'_1, \boldsymbol{u}'_2, \boldsymbol{u}'_3\}$ of $\mathcal{L}'_1$. Let $\boldsymbol{u}_j$ be the vector of the 1-6th entries consisting of $\boldsymbol{u}'_j$ for $j = 1, 2$ and 3. We then have

$$\begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}_3 \end{pmatrix} = \begin{pmatrix} 1 & 11464 & -3475226 & 80 & 5520 & 916415 \\ 0 & 27025 & -8194204 & 0 & 12000 & 2328055 \\ 0 & 0 & 0 & 125 & 675 & -110438 \end{pmatrix}.$$

By applying the LLL reduction to the lattice $\mathcal{L}_1$ spanned by $\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3$, we have an LLL reduced basis

$$\begin{pmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \boldsymbol{a}_3 \end{pmatrix} = \begin{pmatrix} 1568 & 3927 & -8708 & -435 & -4365 & -6789 \\ -1792 & -4488 & 9952 & 515 & 5085 & -8018 \\ 3841 & 9499 & 15250 & -1095 & -10905 & -1034 \end{pmatrix}.$$

However, actually, the target vector $(\boldsymbol{s}'_1, \boldsymbol{s}'_2)$ defined by the coefficients of s'_1 and s'_2 is

$$\left(\boldsymbol{s}'_1, \boldsymbol{s}'_2\right) = (33, -38, 36398, 15, -15, -31877).$$

Thus $\boldsymbol{a}_i$ does not coincide with both of $(\boldsymbol{s}'_1, \boldsymbol{s}'_2)$ and $-(\boldsymbol{s}'_1, \boldsymbol{s}'_2)$ for any $1 \leq i \leq 3$. Note that 1-2nd and 4-5th entries of the correct $(\boldsymbol{s}'_1, \boldsymbol{s}'_2)$ are much smaller than its other entries. This is true in many cases from the constructions of X, s'_1 and s'_2 described in Sect. 3 of [25] and Sect. 4.2 in this paper. On the other hand, the absolute values of all entries of $\boldsymbol{a}_i$ have almost the same sizes for $1 \leq i \leq 3$. Moreover, it is easy to see $\left\|(\boldsymbol{s}'_1, \boldsymbol{s}'_2)\right\|_p > \max\{\|\boldsymbol{a}_1\|_p, \|\boldsymbol{a}_2\|_p, \|\boldsymbol{a}_3\|_p\}$ for any $1 \leq p \leq \infty$, where $\|\cdot\|_p$ denotes the p-norm. For example, we have $\|(\boldsymbol{s}'_1, \boldsymbol{s}'_2)\|_2 \approx 48383.47 > \max\{\|\boldsymbol{a}_1\|_2, \|\boldsymbol{a}_2\|_2, \|\boldsymbol{a}_3\|_2\} \approx 21418.08$. This means that our target vector $(\boldsymbol{s}'_1, \boldsymbol{s}'_2)$ is not shortest in $\mathcal{L}_1$ of 3-rank in terms of $\|\cdot\|_p$ for any $1 \leq p \leq \infty$. Thus, it seems that the LLL lattice basis reduction in terms of well-known norms , e.g., $\|\cdot\|_p$ for $1 \leq p \leq \infty$, does not work well for finding $(\boldsymbol{s}'_1, \boldsymbol{s}'_2)$.

To obtain $\left(\boldsymbol{s}'_1, \boldsymbol{s}'_2\right)$, we apply the weighted LLL reduction for the weight $\boldsymbol{w}$ described below to $\mathcal{L}_1$ since the above situation is good for the weighted LLL reduction, see Sect. 4 in [14]. Recall that $\boldsymbol{X} = (125, 675, -110438)$. We have

$$\left(\frac{H(X)}{125}, \frac{H(X)}{675}, \frac{H(X)}{110438}\right) = \left(\frac{110438}{125}, \frac{110438}{675}, 1\right).$$

Put

$$\boldsymbol{w} = \left(2^{\left\lfloor \log_2\left(\frac{110438}{125}\right)\right\rfloor}, 2^{\left\lfloor \log_2\left(\frac{110438}{675}\right)\right\rfloor}, 1, 2^{\left\lfloor \log_2\left(\frac{110438}{125}\right)\right\rfloor}, 2^{\left\lfloor \log_2\left(\frac{110438}{675}\right)\right\rfloor}, 1\right)$$
$$= \left(2^9, 2^7, 1, 2^9, 2^7, 1\right).$$

We obtain the following weighted LLL reduced basis of $\mathcal{L}_1^{\boldsymbol{w}}$:

$$\begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \end{pmatrix} = \begin{pmatrix} 33 & -38 & 36398 & 15 & -15 & -31877 \\ -33 & 38 & -36398 & 110 & 690 & -78561 \\ -158 & -637 & 74040 & -15 & 15 & 31877 \end{pmatrix}.$$

Note that $\boldsymbol{b}_1$ just coincides with $\left(\boldsymbol{s}_1', \boldsymbol{s}_2'\right)$.

4.3.2 Second step: obtaining a candidate of *f*

Here, we obtain a candidate of f. We set

$$f := c_1^{(f)} x^3 + c_2^{(f)} y + c_3^{(f)},$$
$$r_j' := c_1^{(j)} x^3 + c_2^{(j)} y + c_3^{(j)} \quad (j = 1 \text{ and } 2),$$

where $c_i^{(f)}$'s and $c_i^{(j)}$'s are indeterminates. By substituting s_1' and s_2' obtained in Step 1 into the equalities (2) and (3), and by comparing the coefficient of $\underline{x}^{\underline{\boldsymbol{i}}}$ for each $\underline{\boldsymbol{i}} \in \Lambda_{X^2}$, we have the linear system $\boldsymbol{v}\boldsymbol{B} = \boldsymbol{b}$, where B is a (9×6) matrix. The rank of the kernel lattice $\mathcal{L}_2$ of $\boldsymbol{B}$ is equal to 1. We obtain a solution $\boldsymbol{v}_0$ to $\boldsymbol{v}\boldsymbol{B} = \boldsymbol{b}$ and a basis $\{\boldsymbol{v}_1\}$ of $\mathcal{L}_2$ as follows:

$$\begin{pmatrix} \boldsymbol{v}_0 \\ \boldsymbol{v}_1 \end{pmatrix} = \begin{pmatrix} -32 & -3804373 & 840328137 & 89131 & -509276 & 275909743 & 26620 & -546123 & -241370517 \\ 125 & 675 & -110438 & -33 & 38 & -36398 & -15 & 15 & 31877 \end{pmatrix}.$$

Compute another solution $\boldsymbol{v}_0' := \boldsymbol{v}_0 - \lfloor \langle \boldsymbol{v}_0, \boldsymbol{v}_1 \rangle / \langle \boldsymbol{v}_1, \boldsymbol{v}_1 \rangle \rceil \boldsymbol{v}_1$ to $\boldsymbol{v}\boldsymbol{B} = \boldsymbol{b}$. Let $\boldsymbol{v}_0''$ be the vector consisting of the 1-3rd entries of $\boldsymbol{v}_0'$. We then have

$$\boldsymbol{v}_0'' = (950468, 1328327, 557585),$$

and set

$$f' := 950468x^3 + 1328327y + 557585.$$

Note that the polynomial f' obtained from $\boldsymbol{v}_0''$ is closer to the correct f than the one obtained from $\boldsymbol{v}_0$. We also note that it is possible to proceed to the next step even if f' does not coincide with f, see Remark 18.

4.3.3 Third Step: Recovery of $\widetilde{m}$

Finally, we recover $\widetilde{m}(x, y)$. We find $s_1(x, y)$ before recovering $\widetilde{m}(x, y)$. Put

$$\begin{aligned}\widetilde{m} &:= c_1x^3 + c_2y + c_3,\\ s_1 &:= c_4x^3 + c_5y + c_6,\\ r_1 &:= c_7x^3 + c_8y + c_9,\end{aligned}$$

where c_i's are indeterminates. By substituting f' obtained in Step 2 into the equalities (5), and by comparing the coefficient of $\underline{x}^{\underline{\boldsymbol{i}}}$ for each $\underline{\boldsymbol{i}} \in \Lambda_{X^2}$, we have the linear system $\boldsymbol{w}\boldsymbol{C} = \boldsymbol{c}$, where $\boldsymbol{C}$ is a (9×6) matrix. The rank of the kernel lattice $\mathcal{L}_3$ of $\boldsymbol{C}$ is equal to 3. We fix a solution $\boldsymbol{w}_0$ to the system and a basis $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3\}$ of $\mathcal{L}_3$ as follows:

$$\begin{pmatrix}\boldsymbol{w}_0\\ \boldsymbol{w}_1\\ \boldsymbol{w}_2\\ \boldsymbol{w}_3\end{pmatrix} = \begin{pmatrix}225204073068 & 315361848743 & -6569529455 & -10 & 249 & 0 & 1640912 & 2687357 & 0\\ 1 & 163580614 & -36132895073 & 0 & 0 & 43 & 0 & 0 & -326961\\ 0 & 475525025 & -105037483109 & 0 & 0 & 125 & 0 & 0 & -950468\\ 0 & 0 & 0 & 125 & 675 & -110438 & -950468 & -1328327 & -557585\end{pmatrix}.$$

We find a vector $\boldsymbol{z}$ in the lattice $\langle \boldsymbol{w}_1, \boldsymbol{w}_2\rangle_{\mathbb{Z}}$ close to $\boldsymbol{w}_0 + \boldsymbol{w}_3$ by applying the Babai nearest plane algorithm. We then have the matrix

$$\begin{pmatrix}225203926700 & 315361701825 & -6569550089 & 0 & 0 & -236775 & 0 & 0 & -1254928\\ 146368 & 146918 & 20634 & 115 & 924 & 126337 & 690444 & 1359030 & 697343\end{pmatrix},$$

where 1st and 2nd rows are the vectors $\boldsymbol{z}$ and $\boldsymbol{w}_0 + \boldsymbol{w}_3 - \boldsymbol{z}$, respectively. The vector embedded in $\boldsymbol{w}_0 + \boldsymbol{w}_3 - \boldsymbol{z}$ as its 4–6th entries is equal to the correct $\boldsymbol{s}_1$.

Next, we compute r satisfying $F_1 - \widetilde{m} - s_1' = rX$. Note that there exists a polynomial r satisfying the above equality, and that we can recover $\widetilde{m}$ if we obtain such an r (cf. Remark 18 and Step 3-4 in Sect. 4.2). We set

$$\begin{aligned}r &:= c_1x^3 + c_2y + c_3,\\ \widetilde{m} &:= c_4x^3 + c_5y + c_6,\end{aligned}$$

where c_i's are indeterminates. In the equality $F_1 - s_1f' = \widetilde{m} + rX$, by comparing the coefficient of $\underline{x}^{\underline{\boldsymbol{i}}}$ for each $\underline{\boldsymbol{i}} \in \Lambda_{X^2}$, we have the linear system $\boldsymbol{x}\boldsymbol{H} = \boldsymbol{h}$, where $\boldsymbol{H}$ is a (6×6) matrix. The rank of the kernel lattice $\mathcal{L}_4$ of $\boldsymbol{H}$ is equal to 1. We fix a solution $\boldsymbol{x}_0$ to $\boldsymbol{x}\boldsymbol{H} = \boldsymbol{h}$ and a basis $\{\boldsymbol{x}_1\}$ of $\mathcal{L}_4$ as follows:

$$\begin{pmatrix}\boldsymbol{x}_0\\ \boldsymbol{x}_1\end{pmatrix} = \begin{pmatrix}2591380 & 4015684 & 0 & 226710493 & 1223593193 & -200170290060\\ 0 & 0 & 1 & -125 & -675 & 110438\end{pmatrix}.$$

We set

$$r' := 2591380x^3 + 4015684y + 1.$$

There exists a unique $t \in \mathbb{Z}$ such that $r = r' + t$. Our aim is to find such an integer t, see Steps 3-3 and 3-4 in Sect. 4.2. Let $\underline{\boldsymbol{k}}$ be the maximal element in Λ_X. Put

$$\begin{aligned}
e' &:= e^{-1} \pmod{\varphi(d)} \\
&= 1, \\
H_1 &:= F_1 - s_1 f' - r' X \\
&= 226710368x^3 + 1223592518y - 200170179622, \\
\mu &:= c_{\underline{\boldsymbol{k}}}(H_1) \\
&= -200170179622, \\
c_{\underline{\boldsymbol{k}}}(m') &:= \mu^{e'} \pmod{d} \quad \left(0 < c_{\underline{\boldsymbol{k}}}(m') < d\right) \\
&= 3, \\
c_{\underline{\boldsymbol{k}}}(\widetilde{m}) &= \left(c_{\underline{\boldsymbol{k}}}(m')\right)^e \pmod{Nd} \quad \left(0 < c_{\underline{\boldsymbol{k}}}(\widetilde{m}) < Nd\right) \\
&= 146243, \\
t &= \left(\mu - c_{\underline{\boldsymbol{k}}}(\widetilde{m})\right) / c_{\underline{\boldsymbol{k}}}(X) \\
&= 1812513, \\
\widetilde{m} &= F_1 - s_1 f' - \left(r' + t\right) X \\
&= 146243x^3 + 146243y + 131072.
\end{aligned}$$

We succeeded in recovering $\widetilde{m}(x, y)$ in Sect. 3.6.

5 Complexity analysis

In this section, we investigate the complexity of the algorithm in Sect. 4.2. We analyse our attack in accordance with the parameter sizes in Sect. 3.5 (cf. Sect. 5 in [25]). Let $X \in \mathbb{Z}[\underline{x}]$ be a public key of DEC. Let w_X and Λ_X denote the total degree and the support of X, respectively. To simplify the notations, we set $w := w_X$, assume $w = \#\Lambda_X$ and fix b, where b is the maximum of the bit length of the coefficients of X except its leading and constant terms. We show that the attack performs in polynomial time in terms of the parameters w and λ. Here note that w and λ are independent of each other, see Remark 7 in Sect. 3. In our complexity analysis, we use the same notation as in Sect. 4.2. The parameters d and e are $O\left(2^{\lambda}\right)$ and $O(w\lambda)$, respectively. Note that the size of each coefficient of F_j is $O(w\lambda)$ bits for $j = 1, 2$ and 3, see Sect. 3.5 for the representation of the parameters by w and λ. Assume that the size of each coefficient of $s_1 f'$ is bounded by $O(w\lambda)$ bits.

Remark 21 First, let us determine the bit complexity of the computation of polynomials with integer coefficients in the algorithm. We suppose the arithmetic operations of addition and subtraction of two polynomials $F, G \in \mathbb{Z}[\underline{x}]$ are $O(\min\{q_F, q_G\})$ in $\mathbb{Z}$, where q_F and q_G are the number of the terms of F and G, respectively. Moreover, the arithmetic operations of multiplication of them are $O\left((\max\{q_F, q_G\})^2\right)$ in $\mathbb{Z}$. We compute ${F_1}' := F_1 - F_2$ and ${F_2}' := F_2 - F_3$ at the beginning of the algorithm. Note that the number of the terms of F_j is at most w^2 for each $1 \leq j \leq 3$. The sizes of the

coefficients of F_j are $O(w\lambda)$ for $j = 1, 2$ and 3. Thus the arithmetic complexity of computing F_1' and F_2' is $O\left(w^2\right)$, and its bit complexity is

$$O\left(w^2 (w\lambda)\right) = O\left(w^3\lambda\right). \tag{10}$$

We do such computations in (6)–(9). The arithmetic complexity of (6)–(9) is $O(w^4)$ and thus the bit complexity is

$$O\left(w^4 (w\lambda)^2\right) = O\left(w^6\lambda^2\right) \tag{11}$$

since the sizes of the coefficients of the polynomials appearing in (6)–(9) are $O(w\lambda)$ bits. Note that we regard the coefficients of certain polynomials as indeterminates. (For example, in (6), we regard the coefficients of $s_1{}'$, $s_2{}'$ and g as indeterminates.) On the other hand, we compute $H_1 := F_1 - s_1 f' - rX$ in Step 3-4. In this case, we do not regard any coefficient as indeterminates. Since for each of s_1, f', r and X, the number of its terms is w, we require $O\left(w^2\right)$ arithmetic operations for computing $s_1 f'$ and rX. In addition, for each of F_1, $s_1 f'$ and rX, the number of its terms is $O(w^2)$. Here recall that the size of each coefficient of the polynomials F_1, $s_1 f'$ and rX is $O(w\lambda)$ bits. Thus the bit complexity of computing H_1 is

$$O\left(w^2 (w\lambda)^2\right) = O\left(w^4\lambda^2\right). \tag{12}$$

Remark 22 Second, we solve one or two linear systems in each step of our attack. Then, we obtain one solution and the kernel lattice for each linear system. We assume that the bit complexity of solving a non-homogeneous linear system is equivalent to the bit complexity of computing the (row) Hermite Normal Form (HNF) of the augmented matrix of the system. According to Chapter 2 in [16], we assume that the computation of the HNF of an $n \times m$ matrix $\boldsymbol{M} = (M_{i,j})_{i,j}$ requires $O(nm^4(\log(\|\boldsymbol{M}\|_\infty))^2)$ bit operations, where $\|\boldsymbol{M}\|_\infty := \max_{i,j}\{|M_{i,j}|\}$. On the other hand, we assume that a homogeneous linear system is solved by the Gaussian elimination.

To simplify the notations, we assume the sizes of the entries of one solution and an output basis of the kernel lattice of each linear system are $O(\ell)$ bits if the sizes of the entries of its augmented matrix are $O(\ell)$ bits.

Remark 23 Third, we discuss the size of the norm of a vector with integer entries. Let $\boldsymbol{a} = (a_1, \ldots, a_k) \in \mathbb{Z}^k$ be a vector with $|a_i| \leq 2^l$ for $1 \leq i \leq k$. Since $\|\boldsymbol{a}\| \leq \sqrt{k2^{2l}}$, the size of $\|\boldsymbol{a}\|$ is bounded by $\log\left(\sqrt{k2^{2l}}\right) = \log\left(k^{1/2}\right) + l = O(\log(k) + l)$ bits. Similarly, the size of $\|\boldsymbol{a}\|^2$ is $O(\log(k) + l)$ bits.

5.1 The complexity of first step

Step 1-1 We estimate the bit complexity for solving the linear system $\boldsymbol{u}\boldsymbol{A} = \boldsymbol{0}$ with at most $2w + w^2$ indeterminates and w^3 equations. Since this linear system is homo-

geneous, the arithmetic complexity in $\mathbb{Z}$ of solving the linear system is $O\left(w^6\right)$, see Remark 22. The size of each entry of $\boldsymbol{A}$ is $O\left(w\lambda\right)$ bits, and thus Step 1-1 requires

$$O\left(w^8\lambda^2\right) \tag{13}$$

bit operations. In addition, we note that the sizes of the entries of $\boldsymbol{u}'_1$, $\boldsymbol{u}'_2$ and $\boldsymbol{u}'_3$, that are basis vectors of the kernel lattice $\mathcal{L}'_1$ of $\boldsymbol{A}$, are $O\left(w\lambda\right)$ bits from Remark 22.

Step 1-2 In the beginning of this step, we compute $\boldsymbol{UW}$, where $\boldsymbol{U}$ is a basis matrix of $\mathcal{L}_1$ with $3 \times 2w$ entries and $\boldsymbol{W}$ is a $(2w \times 2w)$ diagonal matrix. The arithmetic complexity of multiplying these matrices is $3 \times (2w) = O\left(w\right)$. Since the size of each entry of $\boldsymbol{U}$ and $\boldsymbol{W}$ is $O\left(w\lambda\right)$ bits, the multiplying runs in

$$O\left(w \times (w\lambda)^2\right) = O\left(w^3\lambda^2\right) \tag{14}$$

bit operations. We note that the size of each entry of $\boldsymbol{UW}$ is $O\left(w\lambda\right)$ bits. After the multiplying, we execute the LLL reduction to the $2w$-dimensional lattice $f_{\boldsymbol{W}}\left(\mathcal{L}_1\right)$ of 3-rank with the basis matrix $\boldsymbol{UW}$. According to [19], the computation of the LLL reduction requires

$$O\left(3^5\left(2w\right)\left(\log\left(2w \times 2^{2w\lambda}\right)\right)^3\right)$$

bit operations in this case because the norms of the row vectors of $\boldsymbol{UW}$ are $O\left(\sqrt{2w \times 2^{2w\lambda}}\right)$. Thus the LLL reduction of this step runs in

$$O\left(w^4\lambda^3\right) \tag{15}$$

bit operations. Th size of any entry of the vectors of the LLL reduced basis is $O\left(\sqrt{3\left(w \times 2^{2w\lambda}\right)}\right)$ because the rank of $f_{\boldsymbol{W}}\left(\mathcal{L}_1\right)$ is equal to 3, and because $\|\boldsymbol{u}_i\boldsymbol{W}\|^2 = O\left(w \times 2^{2w\lambda}\right)$ for $i = 1, 2$ and 3, and row vectors $\boldsymbol{u}_i$ of $\boldsymbol{U}$. Thus the size of any entry of the LLL-reduced basis matrix of $\boldsymbol{UW}$ is $O\left(w\lambda\right)$ bits. We multiple the diagonal matrix $\boldsymbol{W}^{-1}$ by the LLL reduced basis matrix. The arithmetic complexity of the multiplying is $3 \times 2w = O\left(w\right)$. Thus the multiplying runs in

$$O\left(w\left(w\lambda\right)^2\right) = O\left(w^3\lambda^2\right) \tag{16}$$

bit operations.

5.2 The complexity of second step

Step 2-1 In this step, we solve the linear system $\boldsymbol{vB} = \boldsymbol{b}$ with $3w$ indeterminates and at most w^2 equations. From Remark 22, the bit complexity of this step can be estimated as

$$O\left(w^{11}\lambda^2\right). \tag{17}$$

Every entry of a solution and basis vectors of the kernel lattice $\mathcal{L}_2$ has the size of $O(w\lambda)$ bits from the same reason as Step 1-1. Note that $\mathcal{L}_2$ is a $3w$-dimensional lattice of 1-rank. Hence the sizes of the norms of $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ are $O\left(\sqrt{(3w \times 2^{2w\lambda})}\right)$.

Step 2-2 In this step, we compute $\boldsymbol{v}_0' := \boldsymbol{v}_0 - \lfloor \langle \boldsymbol{v}_0, \boldsymbol{v}_1 \rangle / \langle \boldsymbol{v}_1, \boldsymbol{v}_1 \rangle \rceil \boldsymbol{v}_1$. This computation requires $O\left(2^4 (3w)\left(\log\left(3w \times 2^{2w\lambda}\right)\right)^2\right)$ bit operations in accordance with Chapter 17 in [16]. Hence Step 2-2 requires

$$O\left(w^3\lambda^2\right) \tag{18}$$

bit operations.

5.3 The complexity of third step and the total complexity of our attack

Step 3-1 In this step, we compute a solution to the system $\boldsymbol{w}\boldsymbol{C} = \boldsymbol{c}$, and a basis of the kernel lattice of $\boldsymbol{C}$ with $3w$ indeterminates and at most w^2 equations. In a similar way to Step 2-1, the computation requires

$$O\left(w^{11}\lambda^2\right) \tag{19}$$

bit operations. Every entry of a solution $\boldsymbol{w}_0$ and basis vectors $\boldsymbol{w}_1$, $\boldsymbol{w}_2$ and $\boldsymbol{w}_3$ of the kernel lattice $\mathcal{L}_3$ has the size of $O(w\lambda)$ bits. Note that the norms of $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$ and $\boldsymbol{w}_3$ are $O\left(\sqrt{(3w \times 2^{2w\lambda})}\right)$.

Step 3-2 In this step, we apply the Babai nearest plane algorithm to the $3w$-dimensional lattice $\mathcal{L}_3' := \langle \boldsymbol{w}_1, \boldsymbol{w}_2 \rangle$ of 2-rank and the vector $\boldsymbol{w}_0 + \boldsymbol{w}_3$. Before executing the Babai nearest plane algorithm, we execute the LLL reduction to $\mathcal{L}_3'$. Since $\mathcal{L}_3'$ has 2-rank and $3w$-dimension, the LLL reduction requires

$$O\left(2^5 (3w)\left(\log\left(3w \times 2^{2w\lambda}\right)\right)^3\right)$$

bit operations. Thus the LLL reduction in Step 3-2 requires $O\left(w^4\lambda^3\right)$ bit operations. The norm of any vector of the LLL reduced basis is

$$O\left(\sqrt{2\left(3w \times 2^{2w\lambda}\right)}\right)$$

(cf. Chapter 17 in [16]). In a similar way to deriving the bit complexity of Gram-Schmidt algorithm (see Theorem 17.3.4 in [16]), one can verify that the Babai

nearest plane algorithm requires $O\left((3w)^5 2^3 \left(\log\left(\sqrt{2\left(3w \times 2^{2w\lambda}\right)}\right)\right)^2\right)$ bit operations. From this, the bit complexity of the Babai nearest plane algorithm is $O\left(w^7\lambda^2\right)$ in this case. Hence Step 3-2 runs in

$$O\left(w^4\lambda^3\right) + O\left(w^7\lambda^2\right) \tag{20}$$

bit operations.

Step 3-3 We compute a solution to $\boldsymbol{x}\boldsymbol{H} = \boldsymbol{h}$ with $2w$ indeterminates and at most w^2 equations. The size of any entry of $\boldsymbol{H}$ and $\boldsymbol{h}$ is $O(w\lambda)$ bits. Hence Step 3-3 runs in

$$O\left(w^{11}\lambda^2\right) \tag{21}$$

bit operations.

Step 3-4 At the beginning of this step, we compute $e' := e^{-1} \bmod \varphi(d)$ by using the extended Euclid's algorithm. According to Remark 3.5 in [25], the integer d should be chosen so that one can compute $\varphi(d)$ efficiently because the computation is needed in the decryption process (see [25], Sect. 3.4). In Remark 3.5 of [25], the integer d is expected to be a prime number as such an example. From this, we assume d is a prime number, and then we have $\varphi(d) = d - 1$.

Next, we compute $c_{\underline{k}}(m') := \mu^{e'} \pmod d$ $\left(0 < c_{\underline{k}}(m') < d\right)$, where $e' := e^{-1} \pmod{\varphi(d)}$ and μ is a certain coefficient of $H_1(\underline{x})$ (cf. Step 3-4 in Sect. 4.2). Recall that the bit sizes of e', μ and d are $O(\lambda)$, $O(w\lambda)$ and $O(\lambda)$, respectively. Thus this computation can be done in $O\left(w\lambda^2 + \lambda^3\right)$ bit operations by the square-and-multiply algorithm for modular exponentiation.

Third, we compute $c_{\underline{k}}(\widetilde{m}) := \left(c_{\underline{k}}(m')\right)^e \pmod{Nd}$ $\left(0 < c_{\underline{k}}(\widetilde{m}) < Nd\right)$. Recall from Sect. 3.5 that the size of N is $O(w\lambda)$ bits. Note that the sizes of $c_{\underline{k}}(m')$, e and Nd are $O(\lambda)$, $O(\log(w\lambda))$ and $O(w\lambda)$ bits, respectively. Thus, the square-and-multiply algorithm requires $O\left((w\lambda)^2 \log(w\lambda)\right)$ bit operations to compute $c_{\underline{k}}(\widetilde{m})$. As a consequence, those modular exponential arithmetic can be performed in $O\left(\lambda^3 + w^2\lambda^2 \log(w\lambda)\right)$ bit operations. Finally, the computation of $t := \left(\mu - c_{\underline{k}}(\widetilde{m})\right)/c_{\underline{k}}(X)$ runs in $O\left(w^2\lambda^2\right)$ bit operations. The total bit complexity of Step 3-4 is

$$O\left(\lambda^3 + w^2\lambda^2 \log(w\lambda)\right). \tag{22}$$

Putting all the steps together, namely considering (10)–(22), we can determine the complexity of our attack.

Theorem 24 *The total bit complexity of the attack in Sect. 4.2 is*

$$O\left(w^{11}\lambda^2\right) + O\left(w^4\lambda^3\right).$$

Consequently, our attack performs in polynomial time for all the parameters λ and w_X, where λ and w_X are independent of each other.

Remark 25 The estimated complexity in Theorem 24 shows that the computation of our attack may become expensive for large $w = w_X$ and $\#\Lambda_X \leq w$. Thus, to secure DEC, one can think of increasing the parameters w and $\#\Lambda_X$. However, DEC is impractical for large w_X and $\#\Lambda_X$ since ciphertexts of DEC have exceedingly large sizes. For example, when $w_X = \#\Lambda_X = 45$, $b = 10$ and $\lambda = 128$, we generated 100 ciphertexts (F_1, F_2, F_3, N) in accordance with Sects. 3.2 and 3.3, and measured their sizes. As a result, their average size is about 10,086,237 bits.

Remark 26 From the above reason, the dominant term of the estimated complexity in Theorem 24 is $O(w^4\lambda^3)$ in practice.

6 Experimental verification

In this section, we demonstrate with experimental results that our attack algorithm enables one to break the one-way property of DEC in practical time. In our experiments, we generated DEC instances of $n = 4$, where n is the number of indeterminates of a public key $X(\underline{x})$. The PC used in our experiments is as follows: The OS is Mac OS X, 64 bit. The processor is 2.60GHz CPU (Intel Corei5). The memory is 16GB. Authors implemented the attack algorithm over Magma V2.21-3 [8]. For the parameters, we adopted recommended ones in Remark 9 (such parameters shall make DEC instances $\lambda = 128$ bit level secure).

Procedures of Our Experiments For three parameters $w_X, \#\Lambda_X$ and b, each of which is independent of the security parameter λ, we conduct the following procedure 100 times:

1. Construct secret/public keys in accordance with Section 3.2.
2. With the public key, we generate a ciphertext in accordance with Sect. 3.3.
3. For the above public key and the ciphertext, recover the twisted plaintext by Attack Algorithm given in Sect. 4.2.

In our experiments, we generated each public key X so that its coefficients have b bit sizes except the terms of its maximal degree and constant., i.e., $2^{b-1} \leq |c_{\underline{i}}(X)| < 2^b$ for all $\underline{\boldsymbol{i}} \in \Lambda_X$ with $\underline{\boldsymbol{i}} \neq \underline{\boldsymbol{k}}, \underline{\mathbf{0}}$. Here $\underline{\boldsymbol{k}}$ denotes the maximal element of Λ_X, see Remark 4 (2) for the ordering. For each DEC instance generated as above, we also apply a variant of Attack Algorithm in order to show the effectiveness of weighted LLL reduction for our cryptanalysis. Here the variant adopts the LLL reduction in terms of the Euclidean norm in the first step of the original attack instead of the weighted LLL reduction. We measure the number of successes and time performance only if our attack succeeds, i.e., $\widetilde{m}$ or $-\widetilde{m}$ is recovered in the final step.

Table 1 indicates results of our experiments on our cryptanalysis of DEC instances. In Step 1 of the table, the number of successes is shown only if the target lattice point $(\boldsymbol{s}'_1, \boldsymbol{s}'_2)$ or $-(\boldsymbol{s}'_1, \boldsymbol{s}'_2)$ is found. For the target lattice point, see Step 1 of Attack Algorithm in Sect. 4.2. In Step 3 of the table, the number of successes is shown only if we succeeded in finding a twisted plaintext $\widetilde{m}$ (or $-\widetilde{m}$).

We see from the results of Step 1 in Table 1 that the weighted LLL reduction recovered our target lattice point in Step 1 with high probability, being about from 70 to 90%. On the other hand, we could not find the target lattice point with the usual LLL reduction in any case of our experiments (we omit to show the experimental results on the attack with the usual LLL reduction). We see from the results of Step 3 in Table 1 that our attack algorithm with the weighted LLL reduction could find the twisted plaintext $\widetilde{m}$ (or $-\widetilde{m}$) with sufficiently high probability, being about from 20 to 40%. We, however, could not succeed in finding the twisted plaintext at all by another one with the usual LLL reduction. From this, we infer that to adopt the weighted LLL reduction is quit important for our attack to succeed, and that our attack with the weighted LLL reduction has sufficiently high success probability for practical cryptanalysis.

From the point of view on the efficiency of generating keys and encryption/decryption, we consider that the parameters of Table 1 are practical. We also refer to Tables 4, 5 and 6 given in Sect. 6 of the designer's paper [25]. We conclude from these experimental results that the attack algorithm can break, with sufficiently high probability, the one-way property of DEC in practical time.

Remark 27 The ranks of lattices occurring in Step 1 are equal to 3 in many cases. In fact, this is true for 100 instances of DEC constructed in our experiments. The LLL reduction finds shortest vectors in such lattices of low rank with high probability. In Step 1, a weighted norm is determined so that the target vector becomes a (nearly) shortest vector in terms of the norm. Thus the most important vector for our attack (the target vector in Step 1) is found by the weighted LLL reduction with high probability.

Remark 28 The existence of some failures of our attack suggests that there may exist a method to resist our attack. We analyzed some failure cases and found a reason why our attack failed in finding target lattice points in Steps 1 and 3. In Step 1 of each failure case, the weighted LLL algorithm found a shortest vector, but our target lattice point was not shortest. Similarly, in Step 3 of each failure case, our target lattice point was not a closest vector, while the Babai nearest plane algorithm found a closest vector. Therefore one may resist our attack if it is possible to choose random polynomials or public/secret keys such that our target lattice points are not shortest or closest in lattices ocurring in Steps 1 and 3. However, special choices of polynomials may lead us to another attack, and adding brute force methods to our attack seems to find target lattices points in such cases (see below). In order to resist our attack, we conclude that a major improvement of DEC is required. For example, the number of ciphertexts (polynomials) should be reduced from 3 to 2 or 1 because using 3 ciphertexts is essential to our attack.

On the other hand, we consider whether there is room for improving our attack or not. A simple improvement is to add steps of brute force search (with small range) to Steps 1 and 3. Our analysis in Sect. 4.1 suggests that our target vectors in Steps 1 and 3 are nearly shortest and nearly closest vectors, respectively, and thus our target vectors seem to be found by brute force methods with small range. However, we omit to conduct experiments on our attack with brute force methods. As mentioned above, we believe that our attack has already provided a practical solution to a problem of breaking DEC which is a candidate of PQC with small key sizes.

Table 1 Experimental results on Attack Algorithm given in Sect. 4.2 for DEC of 128 bit level security with 4 indeterminates

Parameters recommended in Sect. 3.5		Experimental results		
Value of b	Number of monomials of X	Number of successes of (1st and 3rd Steps in) the attack / 100		
		1st Step	3rd Step	Ave. Time (s)
10	3	71	23	0.02
	4	81	33	0.03
	5	86	29	0.04
	6	86	35	0.06
	7	85	29	0.07
	8	92	33	0.09
	9	88	41	0.21
	10	91	32	0.25
50	3	68	21	0.02
	4	82	39	0.03
	5	77	30	0.04
	6	83	33	0.07
	7	88	41	0.08
	8	93	32	0.10
	9	92	36	0.21
	10	91	34	0.28
100	3	75	29	0.02
	4	78	26	0.03
	5	80	36	0.04
	6	83	31	0.07
	7	82	34	0.08
	8	95	40	0.11
	9	87	36	0.20
	10	91	38	0.27

The total degree of each public key is $w_X = 10$. The authors conducted experiments in accordance with Procedures of Our Experiments, which is given at the first part of Sect. 6. The parameter b is the bit length of the coefficients of a public key X except the terms of its maximal degree and constant. "Ave. Time" denotes the average of time which it took to perform the attack algorithm. (Time performance is shown for successful cases.)

7 Conclusion

We present in this paper a polynomial time-attack based on the weighted LLL reduction against the one-way property of a Diophantine Equation-based Cryptosystem (DEC), which was proposed in 2015 by the third author of this paper as one of the candidates of Post-Quantum Cryptosystems (PQC). Compared with other well-known candidates

of PQC, sizes of public keys in DEC are much smaller, e.g., about 1,200 bits for 128 bit level security. This is a strongly desired characteristic for candidates of PQC.

Diophantine equations are generally unsolvable, and thus it is expected to be a base of the security of PQC. However, we showed that DEC's security does not rely on the computational hardness to solve Diophantine equations, and that moreover DEC is no longer secure. Concretely, with linearlization technique, one can reduce breaking the one-way property of DEC to computing certain (comparatively) shorter points in low rank-lattices. Our most crucial target lattice point has the following special property: it is not necessarily a shortest lattice point whereas most of the entries are comparatively small. In our attack, even with the LLL reduction in terms of well-known norms, e.g., p-norms for $1 \leq p \leq \infty$, one seems to fail in finding such lattice points.

The most (heuristically-)technical point in our attack is changing the norm in the LLL reduction from the Euclidean norm to an appropriate weighted one. One can see from our analysis that the most important target lattice point becomes a (nearly) shortest lattice point in terms of a weighted norm, where the weight is determined by our heuristic method. Furthermore, the most important target lattice point is embedded in a (weighted) lattice of 3-rank, which implies the weighted LLL reduction can output with high probability such a target point. From this, we applied the weighted LLL reduction, which is the LLL reduction in terms of a weighted norm to our cryptanalysis. Our experimental results and complexity analysis suggest that for all the recommended parameters, the one-way property of DEC can be broken with sufficiently high probability by our polynomial time-attack based on the weighted LLL reduction.

We also demonstrated with our experimental results that the weighted LLL reduction gives an effective computational tool to find lattice points of special characteristic: the sizes of entries are almost known and most of them are small. Hence the weighted LLL reduction can provide a tool to investigate the security of cryptosystems whose security are transformed to the problem of computing such lattice points.

Acknowledgements The authors deeply thank Shun'ichi Yokoyama for many helpful comments, corrections, suggestions on this research, and discussions in the implementations on Magma. The authors also thank Steven Galbraith for helpful comments on Coppersmith's method, and thank Masaya Yasuda for helpful comments on the weighted LLL reduction and corrections on this paper. This work was supported by JST CREST Grant Number JPMJCR14D6, Japan. The authors are grateful to the anonymous referees for their careful reading of our manuscript and their valuable comments and suggestions.

References

1. Akiyama, K., Goto, Y.: An algebraic surface public-key cryptosystem. IEICE Tech. Rep. **104**(421), 13–20 (2004)
2. Akiyama, K., Goto, Y.: A Public-key Cryptosystem using Algebraic Surfaces, In: Proceedings of PQCrypto., pp. 119–138, (2006). http://postquantum.cr.yp.to/. Accessed 19 June 2018

3. Akiyama, K., Goto, Y.: An improvement of the algebraic surface public-key cryptosystem. In: Proceedings of 2008 Symposium on Cryptography and Information Security, SCIS 2008, CD-ROM, 1F1-2, (2008)
4. Akiyama, K., Goto, Y., Miyake, H.: An algebraic surface cryptosystem. In: Jarecki, S., Tsudik, G. (eds.) Public Key Cryptography – PKC 2009. PKC 2009. Lecture Notes in Computer Science, vol. 5443. Springer, Berlin, Heidelberg (2009)
5. Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica **6**(1), 1–13 (1986). (Preliminary version in STACS 1985)
6. Bérczes, A., Hajdu, L., Hirata-Kohno, N., Kovács, T., Pethö, A.: A key exchange protocol based on Diophantine equations and S-integers. JSIAM Lett. **6**, 85–88 (2014)
7. Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.): Post-Quantum Cryptography. Springer-Verlag, Berlin (2009)
8. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. J. Symb. Comput. **24**(3–4), 235–265 (1997)
9. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. J. Cryptol. **10**(4), 233–260 (1997). Springer-Verlag
10. Cusick, T.W.: Cryptoanalysis of a public key system based on diophantine equations. Inf. Process. Lett. **56**(2), 73–75 (1995)
11. Ding, J., Gower, J.E., Schmidt, D.S.: Multivariate public key cryptosystems, advances in information security, 25. Springer, Berlin, Heidelberg (2006)
12. Davis, M., Matijasevič, Y., Robinson, J.: Hilbert's tenth problem, Diophantine equations: positive aspects of a negative solution, mathematical developments arising from Hilbert problems In: Browder, F.E. (ed.) Proceedings of Symposia in Pure Mathematics, vol. 28, pp. 1–34. American Mathematical Society, Providence (1976)
13. Eisenträger, K.: Hilbert's tenth problem for function fields of varieties over number fields and p-adic fields. J. Algebra **310**(2), 775–792 (2007)
14. Faugère, J. -C., Goyet, C., Renault, G.: Attacking (EC)DSA given only an implicit hint. In: Knudsen, L.R., Wu, H. (eds.) Selected Areas in Cryptography. SAC 2012. Lecture Notes in Computer Science, vol. 7707. Springer, Berlin, Heidelberg (2012)
15. Faugère, J. -C., Spaenlehauer, P. -J.: Algebraic cryptanalysis of the PKC'2009 algebraic surface cryptosystem. In: Nguyen, P.Q., Pointcheval, D. (eds.) Public Key Cryptography – PKC 2010. PKC 2010. Lecture Notes in Computer Science, vol. 6056. Springer, Berlin, Heidelberg (2010)
16. Galbraith, S.D.: Mathematics of public key cryptography. Cambridge University Press, New York (2012)
17. Hirata-Kohno, N., Pethö, A.: On a key exchange protocol based on Diophantine equations. Infocommunications J. **5**(3), 17–21 (2013). **(Scientific Association for Infocommunications (HTE))**
18. Iwami, M.: A reduction attack on algebraic surface public-key cryptosystems, lecture notes in computer science, vol. 5081, pp. 323–332. Springer, Berlin (2008)
19. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. **261**(4), 515–534 (1982). **(Springer-Verlag)**
20. Lin, C.H., Chang, C.C., Lee, R.C.T.: A new public-key cipher system based upon the diophantine equations. IEEE Trans. Comput. **44**(1), 13–19 (1995). IEEE Computer Society Washington, DC, USA
21. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. EUROCRYPT 2010. Lecture Notes in Computer Science, vol. 6110. Springer, Berlin, Heidelberg (2010)
22. Misoczki, R., Tillich, J.P., Sendrier, N., Barreto, P.S.L.M.: MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes, Information Theory Proceedings. (ISIT), IEEE International Symposium on Information Theory (2013)
23. A draft of the report on post-quantum cryptography NISTIR 8105. http://csrc.nist.gov/publications/drafts/nistir-8105/nistir_8105_draft.pdf. Accessed 19 June 2018
24. The slides of NIST's announcement "Post-Quantum Cryptography: NIST's Plan for the Future". https://pqcrypto2016.jp/data/pqc2016_nist_announcement.pdf. Accessed 19 June 2018
25. Okumura, S.: A public key cryptosystem based on diophantine equations of degree increasing type. Pac. J. Math. Ind. **7**(4), 33–45 (2015). **(Springer, Berlin Heidelberg)**
26. Pheidas, T.: Hilbert's tenth problem for fields of rational functions over finite fields. Invent. Math. **103**(1), 1–8 (1991). **(Springer-Verlag)**

27. Tao, C., Diene, A., Tang, S., Ding, J.: Simple matrix scheme for encryption. In: Gaborit, P. (ed.) Post-Quantum Cryptography. PQCrypto 2013. Lecture Notes in Computer Science, vol. 7932. Springer, Berlin, Heidelberg (2013)
28. Uchiyama, S., Tokunaga, H.: On the Security of the Algebraic Surface Public-key Cryptosystems (in Japanese), In: Proceedings of 2007 Symposium on Cryptography and Information Security, SCIS 2007, CD-ROM, 2C1-2, (2007)
29. Videla, C.R.: Hilbert's tenth problem for rational function fields in characteristic 2. Proc. Am. Math. Soc. **120**(1A), 249–253 (1994). **(American Mathematical Society)**
30. Voloch, F.: Breaking the akiyama-goto cryptosystem, contemporary mathematics. Arith. Geom. Cryptogr. Coding Theory **487**, 113–118 (2007). **(American Mathematical Society, Providence, RI)**
31. Yosh, H.: The key exchange cryptosystem used with higher order diophantine equations. Int. J. Netw. Secur. Appl. J. **3**(2), 43–50 (2011)

Affiliations

Jintai Ding[1] · Momonari Kudo[2,3,4] · Shinya Okumura[5,6] · Tsuyoshi Takagi[6,7] · Chengdong Tao[8]

✉ Momonari Kudo
m-kudo@math.kyushu-u.ac.jp

Shinya Okumura
okumura@comm.eng.osaka-u.ac.jp

1 Department of Mathematical Sciences, University of Cincinnati, 2600 Clifton Ave, Cincinnati, OH 45220, USA

2 Kobe City College of Technology, 8-3, Gakuen-Higashimachi, Nishi-ku, Kobe 651-2194, Japan

3 Institute of Mathematics for Industry, Kyushu University, 744 Motooka Nishi-ku, Fukuoka 819-0395, Japan

4 Research conducted while at Graduate School of Mathematics, Kyushu University, 744 Motooka Nishi-ku, Fukuoka 819-0395, Japan

5 Department of Information and Communications Technology, Osaka University, 2-1 Yamadaoka Suita, Osaka 565-0871, Japan

6 Research conducted while at Institute of Mathematics for Industry, Kyushu University, 744 Motooka Nishi-ku, Fukuoka 819-0395, Japan

7 Department of Mathematical Informatics, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

8 South China University of Technology, 381 Wushan Rd, Tianhe, 510641 Guangzhou, Guangdon, China