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

Jintai Ding[*] Momonari Kudo[†] Shinya Okumura[‡]
Tsuyoshi Takagi[‡] Chengdong Tao[§]

April 7, 2016

Abstract

Post-quantum cryptography now plays a central role in cryptography. Many candidates of post-quantum cryptosystems (PQC) have been already proposed but require public keys of large sizes. Constructing PQC with public keys of small sizes is strongly desired. In [Oku15], Okumura proposed a public key cryptosystem based on the difficulty of solving Diophantine equations of degree increasing type (DEC for short). DEC is proposed as an analogue of the Algebraic Surface Cryptosystem [AGM09]. DEC has been expected to avoid the analogues of all attacks against ASC (and the previous versions of ASC). Moreover, DEC has been expected to be a candidate of PQC and to achieve the high security with public keys of small sizes, e.g., about 1,200 bits with 128 bit security.

In this paper, we propose a polynomial time attack against DEC. We show that the security of DEC depends on the difficulty of finding special (relatively) short vectors in some lattices obtained from a public key and a ciphertext. The most important target vector in our attack is not necessarily a shortest vector in a lattice of low rank but only some entries are relatively small. In our attack, the LLL algorithm with respect to well-known norms such as the p-norms ($1 \leq p \leq \infty$) does not seem to work well for finding such vectors. The most technical point of our method is to heuristically find a special norm, which we call a weighted norm, such that the most important target vector becomes a (nearly) shortest vector in a lattice of low rank. We call the LLL algorithm with respect to a weighted norm the *"weighted LLL algorithm"* in this paper. Our experimental results by a standard PC with Magma suggest that our attack via the weighted LLL algorithm can break the one-wayness of DEC for 128 bit security proposed in [Oku15] with sufficiently high probability.

Key words— Weighted LLL reduction, Public-key cryrtosystem, Post-quantum cryptosystem, Diophantine equation

[*]Department of Mathematical Sciences, University of Cincinnati.
[†]Graduate School of Mathematics, Kyushu University. E-mail: m-kudo@math.kyushu-u.ac.jp
[‡]Institute of Mathematics for Industry, Kyushu University. E-mail: s-okumura@imi.kyushu-u.ac.jp
[§]South China University of Technology.

Contents

1 Introduction

Post-quantum cryptography is now receiving a great amount of attention. Moreover, it is announced that National Security Agency [NSA] is planning for transitioning to quantum resistant algorithms, and National Institute of Standards and Technology [NIST] published a draft of the report on post-quantum cryptography NISTIR 8105. Many candidates of post-quantum cryptosystems (PQC) have been proposed up to the present, e.g., lattice-based cryptosystems, code-based cryptosystems and multivariate public key cryptosystems (see [BBD08, DGS06] for details). However, these PQC require public keys of large sizes, and thus it is very important task that we find computationally-hard problems for constructing PQC with public keys of small sizes.

The Diophantine problem is well-known to be a hard problem (see [DMR76]) and has been expected to be used to construct PQC. (The Diophantine problem here means that given multivariate polynomials with integer coefficients, find common integral or rational zeros of them.) In fact, a public key cryptosystem [LCL95] and key exchange protocols [BHHKP14, HP13, Yos11] based on the difficulty of the Diophantine problem have been already proposed. However, the one-wayness of the cryptosystem in [LCL95] is broken (see [Cus95]), and the protocols in [BHHKP14, HP13, Yos11] are said to be impractical (see Proposition 2 in [HP13]).

The Diophantine problem is generalized to problems over arbitrary rings. The Algebraic Surface Cryptosystem (ASC) proposed in [AGM09] is based on the difficulty of the Diophantine problem over global function fields which is proved to be unsolvable in general [Phe91, Vid94]. In [AGM09], it is analyzed that we may use public keys of sizes of about only 500 bits. However, the one-wayness of ASC is broken by the ideal decomposition attack proposed in [FS10].

In [Oku15], Okumura proposed a public key cryptosystem based on the difficulty of solving Diophantine equations of degree increasing type over $\mathbb{Z}$ (see Section 3 in this paper for the definition of a polynomial of degree increasing type). We refer to the cryptosystem as DEC for short. In Remark 3.2 of [Oku15], Okumura shows that Diophantine equations of degree increasing type are generally unsolvable. DEC is proposed as a number field analogue of ASC and a candidate of PQC. The main ideas to avoid the analogues of all attacks [FS10, Iwa08, UT07, Vol07] against ASC (and the previous versions [AG04, AG06, AG08] of ASC) are to twist a plaintext by using some modular arithmetic and to use some random polynomials with large coefficients in the encryption process. In Section 4 of [Oku15], it is analyzed that by the above ideas, the number of possible parameters increases, and thus finding the correct plaintext will become infeasible. In addition, the reason why polynomials of degree increasing type are adopted in DEC is to decode a plaintext uniquely even if the plaintext is twisted. In Section 3, we give a brief review of DEC and the recommended parameters for DEC.

Another advantage of DEC is that sizes of public keys are small, e.g., about 1,200 bits with 128 bit security (see Remark 3.5.1), as desired in post-quantum cryptography. Note that well-known efficient candidates of PQC [LPR10, MTSB13, TSTD13] require public keys whose sizes are about 10 times larger than 1,200 bits to achieve 128 bit security. Thus it is important to analyze the security of DEC.

1.1 Our Contribution

In this paper, we propose an attack against DEC. We show that the one-wayness of DEC can be transformed to a problem of finding special relatively short vectors in lattices obtained from a public key and a ciphertext. Our attack can be divided roughly into three steps. In each step, we have a linear system and need to find its appropriate solution, which is equivalent to a problem of finding an appropriate vector in the lattice obtained by solving 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 Babai's nearest plane algorithm [Bab86] and some modular arithmetic.

Our experimental results on our attack in Section 6 show that finding a correct solution in the first step allows us to break 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 key point of our attack is whether the first step succeeds or not.

The lattice obtained in the first step of our attack has low rank, (e.g., 3-rank in many cases), and thus finding a target vector in the lattice seems to be performed by basis reduction algorithms such as the LLL algorithm [LLL82]. However, in Section 4.3, we show an example that the usual LLL algorithm fails in finding the target vector in the first step. Our heuristic analysis on the failure of the example is as follows: the target vector is not necessarily shortest in the lattice of low rank but only some entries are relatively small, i.e., the target vector is a relatively short vector with entries of unbalanced sizes.

1.2 Weighted LLL

In order to deal with such situations, we apply the *weighted LLL algorithm*, which is the LLL algorithm with respect to a special norm called *weighted norm* for some weight, to our attack. We find heuristically a new weighted norm so that the target vector becomes (nearly) shortest in some lattice of low rank with respect to this new norm. By a weighted norm for a vector $\mathbf{a} = (a_1, ..., a_n)$, we mean the norm:

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

where w_i's are positive real numbers, which we call the weight factors. Note that as we will see in Section 4.3, using other well-known norms, e.g., the p-norms $(1 \leq p \leq \infty)$ $\|\mathbf{a}\|_p := (|a_1|^p + \ldots + |a_n|^p)^{\frac{1}{p}}$, in the LLL algorithm does not seem to be effective in finding the target vector. Our method can be also viewed as changing the scale of a lattice to carefully control the entries of a LLL reduced basis of the lattice. Such a method has been used in Coppersmith's method [Cop97] (see also Chapter 19 of [Gal12]) and in [FGR13]. In particular, we consider 2-power integers as w_i $(1 \leq i \leq n)$ to use the knowledge of the bit length of our target vector, such as [FGR13].

1.3 Experimental Verification of Our Attack

Our experimental results in Section 6 show that one can find correct vectors in the first step with probability being about from 70 to 90% for the recommended parameters in Section 3 via the weighted LLL algorithm. Thus we consider that the weighted LLL is effective in cryptanalysis of cryptosystems whose securities are reduced to finding vectors with special properties: they are not shortest, but the bit length of their entries are almost known and relatively small. Moreover, experimental results also show that one can break the one-wayness of DEC with probability being about from 20 to 40%. From this and complexity analysis on our attack, we infer that our attack can break DEC with sufficiently high probability in polynomial time for all practical parameters.

This paper is organized as follows: In Section 2, we give the definition of a weighted norm and describe the weighted LLL algorithm. In Section 3, we give a brief review of DEC. In Section 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 Section 5, we analyze the complexity on our attack. In Section 6, we give some experimental results on the attack.

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, the supervisor of the second author, for helpful comments and corrections. This work was supported by CREST, JST.

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 $\underline{i} = (i_1, \ldots, i_n) \in (\mathbb{Z}_{\geq 0})^n$ and $(a_1, \ldots, a_n) \in R^n$, 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 $\underline{a}^{\underline{i}}$, $\underline{x}^{\underline{i}}$ and $\sum \underline{i}$, respectively. We can write any element $f(\underline{x}) = f(x_1, \ldots, x_n) \in R[\underline{x}] \setminus \{0\}$ (sometimes we also write f simply) in a unique way as a sum of terms:

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

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

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

We call $H(f)$ the height of f. In addition, if for a polynomial $f \in \mathbb{Z}[\underline{x}]$, the support $\Lambda_f = \{\underline{i}_1, \ldots, \underline{i}_q\}$ is ordered by the order coming from the lexicographical order on the monomials of f, then we denote the sequence of the ordered coefficients of f by the vector $\mathbf{f} = \left(c_{\underline{i}_1}(f), \ldots, c_{\underline{i}_q}(f)\right)$.

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 $\{\mathbf{b}_1, \ldots, \mathbf{b}_r\}$, let U be an $r \times m$ matrix whose i-th row vector coincides with $\mathbf{b}_i$ for each i. Then we call 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 $\mathbf{v} \in \mathbb{R}^n$, we denote the Euclidean norm of $\mathbf{v}$ by $\|\mathbf{v}\|$. 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 M be an $m \times n$ matrix over $\mathbb{Z}$ and φ_M the homomorphism as additive groups between $\mathbb{Z}^m \to \mathbb{Z}^n$ defined by $\mathbf{v} \mapsto \mathbf{v}M$. Then the kernel of φ_M is a lattice in $\mathbb{R}^m$, and we call it the kernel lattice of M.

2 Description of Weighted LLL Reduction

In this section, we define weighted norms and weighted lattices, and give the weighted LLL algorithm.

2.1 Definition of Weighted Lattice

Basically, a norm on the lattices in $\mathbb{R}^m$ means the p-norm for some $p > 0$ (the 2-norm is the same as the Euclidean norm). On the other hand, a weighted lattice is defined as a lattice endowed with a special norm which we call a weighted norm. The formal definitions of weighted norms and weighted lattices are as follows:

Definition 2.1.1 For a vector $\mathbf{w} = (w_1, \ldots, w_m) \in (\mathbb{R}_{>0})^m$, we define the map $\| \cdot \|_{\mathbf{w}} : \mathbb{R}^m \to \mathbb{R}$ as follows:

$$\|\mathbf{a}\|_{\mathbf{w}} := \sqrt{(a_1 w_1)^2 + \cdots + (a_m w_m)^2} \quad (\mathbf{a} = (a_1, \ldots, a_m) \in \mathbb{R}^m). \tag{2.1.1}$$

Then $\| \cdot \|_{\mathbf{w}}$ is a norm on $\mathbb{R}^m$, and we call it the *weighted norm* for $\mathbf{w}$. We define a *weighted lattice* for $\mathbf{w}$ in $\mathbb{R}^m$ as a lattice endowed with the weighted norm for $\mathbf{w}$. For any lattice $\mathcal{L} \subset \mathbb{R}^m$ and a vector $\mathbf{w} \in (\mathbb{R}_{>0})^m$, we denote $\mathcal{L}$ by $\mathcal{L}^{\mathbf{w}}$ if we consider $\mathcal{L}$ as a weighted lattice for $\mathbf{w}$.

It is easy to show the following lemma.

Lemma 2.1.2 Let $\mathcal{L} \subset \mathbb{R}^m$ be a lattice and $\mathbf{w} = (w_1, \ldots, w_m) \in (\mathbb{R}_{>0})^m$ a vector. We set W as the following diagonal matrix:

$$W := \begin{pmatrix} w_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & w_m \end{pmatrix}. \tag{2.1.2}$$

We define the isomorphism $f_W : \mathbb{R}^m \longrightarrow \mathbb{R}^m$ by $\mathbf{x} \mapsto \mathbf{x}W$. Then the following are equivalent for any $\mathbf{x} \in \mathcal{L}^{\mathbf{w}}$:

(1) The vector $\mathbf{x}$ is a shortest vector in $\mathcal{L}^{\mathbf{w}}$.

(2) The vector $\mathbf{x}W$ is a shortest vector in $f_W(\mathcal{L})$ with respect to the Euclidean norm.

Remark 2.1.3 Lemma 2.1.2 shows that we can find a shortest vector in $\mathcal{L}^{\mathbf{w}}$ if we find a shortest vector in $f_W(\mathcal{L})$ with respect to the Euclidean norm.

2.2 Weighted LLL Reduction

In this subsection, we define a weighted LLL reduced basis and give an algorithm to find such a basis.

Definition 2.2.1 Let $\mathcal{L}$, W and f_W be as in Lemma 2.1.2. We call an ordered basis $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ of $\mathcal{L}$ a *weighted LLL reduced basis* if $f_W(\mathcal{B}) = \{\mathbf{b}_1 W, \ldots, \mathbf{b}_n W\}$ is an LLL reduced basis of $f_W(\mathcal{L})$ with respect to the Euclidean norm.

We give an algorithm to find a weighted LLL reduced basis.

The weighted LLL algorithm
Input: a vector $\mathbf{w}$ and a basis $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ of $\mathcal{L}$.
Output: a weighted LLL reduced basis $\mathcal{B}' = \{\mathbf{b}_1^{'}, \ldots, \mathbf{b}_n^{'}\}$ of $\mathcal{L}$.

(1) Compute the basis $\{\mathbf{b}_1 W, \ldots, \mathbf{b}_n W\}$ of the lattice $f_W(\mathcal{L})$, where W and f_W are as in Lemma 2.1.2.

(2) Compute an LLL reduced basis $\mathcal{B}^{\mathbf{w}} = \{\mathbf{b}_1^{\mathbf{w}}, \ldots, \mathbf{b}_n^{\mathbf{w}}\}$ of $f_W(\mathcal{L})$ with respect to the Euclidean norm.

(3) Compute $\mathbf{b}_i^{'} := \mathbf{b}_i^{\mathbf{w}} W^{-1}$ $(1 \leq i \leq n)$, and return $\mathcal{B}' = \{\mathbf{b}_1^{'}, \ldots, \mathbf{b}_n^{'}\}$.

In our cryptanalysis (Section 4), the most important vector is not necessarily a shortest vector in a lattice of low rank but only some entries are relatively small. In order to find such a vector, we use the weighted LLL algorithm which can carefully control the entries of a weighted LLL reduced basis (see Section 4.3).

Remark 2.2.2 Controlling the entries of a basis output by the LLL algorithm have been used in Coppersmith's method [Cop97] (see also Chapter 19 of [Gal12]) and in [FGR13]. In their method, the scale of a lattice (or equivalently an inner product used in the LLL algorithm) is changed by heuristic ways. Changing the scale of a lattice or an inner product is equivalent to changing a norm from the Euclidean norm to a weighted norm. In particular, our method for choosing a weighted norm is the same as the method in [FGR13] (see Step 1-2 of our algorithm in Section 4.2).

3 Overview of DEC

In this section, we give a brief review of DEC (see Section 3 in [Oku15] for details). As we mentioned in Section 1, DEC has been expected to be one of candidates of post-quantum cryptosystems which has public keys of small sizes, e.g., about 1,200 bits with 128 bit security (see Remark 3.5.1). Note that 1,200 bits is about 10 times smaller than sizes of public keys of efficient candidates of PQC [LPR10, MTSB13, TSTD13].

3.1 Polynomials of Degree Increasing Type

Definition 3.1.1 A polynomial $X(\underline{x}) \in \mathbb{Z}[\underline{x}] \smallsetminus \{0\}$ is *of degree increasing type* if the map

$$\Lambda_X \longrightarrow \mathbb{Z}_{\geq 0} \; ; \; \underline{i} \mapsto \sum \underline{i}$$

is injective, where we recall that $\sum \underline{i} = \sum_{1 \leq k \leq n} i_k$ for $\underline{i} = (i_1, \ldots, i_n)$.

Remark 3.1.2 Let $X(\underline{x})$ be a polynomial in $\mathbb{Z}[\underline{x}] \smallsetminus \{0\}$.

(1) The polynomial $X(\underline{x})$ is of degree increasing type if and only if the total degrees of the monomials of $X(\underline{x})$ are different each other.

(2) If X is of degree increasing type, then the support Λ_X is a totally ordered set by the following order $\succ$: 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 $i_1 + \cdots + i_n > j_1 + \cdots + j_n$.

Throughout this paper, for a polynomial X of degree increasing type, we endow Λ_X with the total order described in Remark 3.1.2 (2).

Example 3.1.3 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 the DEC scheme according to [Oku15]. Note that although in [Oku15] the security parameter is not suggested, here let us set the security parameter λ.

3.2 Key Generation

Secret Key :

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

Public Key :

(1) An integer d such that $\gcd(a_i, d) = 1$ for all $1 \le i \le n$.

(2) An integer e such that $\gcd(e, \varphi(d)) = 1$, where φ is the Euler function.

(3) An irreducible polynomial $X(\underline{x}) \in \mathbb{Z}[\underline{x}]$ of degree increasing type such that $X(\underline{a}/d) = 0$ and $\#\Lambda_X \le w_X$, where Λ_X and w_X denote the support and the total degree of X, respectively.

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

(1) Choose a finite subset $\Lambda \subset (\mathbb{Z}_{\ge 0})^n$ such that $\#\{\sum \underline{i}\ ;\ \underline{i} \in \Lambda\} = \#\Lambda \ge 3$ and $\underline{0} \in \Lambda$, where $\underline{0} := (0, \ldots, 0) \in (\mathbb{Z}_{\ge 0})^n$.

(2) Let $\underline{k}$ be the maximal element in Λ (note that Λ is a totally ordered set with respect to the order described in Remark 3.1.2 (2)). Choose a random non-zero integer $c_{\underline{i}}$ for each $\underline{i} \in \Lambda \smallsetminus \{\underline{k}, \underline{0}\}$. For a choice of $c_{\underline{i}}$, see Remark 3.5.1 (2).

(3) Choose random integers $c_{\underline{k}}$ and $c_{\underline{0}}$ such that

$$c_{\underline{k}}\underline{a}^{\underline{k}} + c_{\underline{0}}d^w \quad = \quad -\sum_{\underline{i} \in \Lambda \smallsetminus \{\underline{k}, \underline{0}\}} c_{\underline{i}}\underline{a}^{\underline{i}}d^{w - \sum \underline{i}}, \tag{3.2.1}$$

where $w := \max\{\sum \underline{i}\ ;\ \underline{i} \in \Lambda\}$.

(4) Set $\Lambda_X := \Lambda$ and $X(\underline{x}) := \sum_{\underline{i} \in \Lambda_X} c_{\underline{i}}\underline{x}^{\underline{i}}$.

For a choice of X and sizes of e, d and a_i $(i = 1, \ldots, n)$, see Section 3.5 below.

Remark 3.2.1 There exist integers $c_{\underline{k}}$ and $c_{\underline{0}}$ such that the equality (3.2.1) is satisfied because a_i and d are mutually prime for each $i \in \{1, \ldots, n\}$ from the assumption.

3.3 Encryption

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

(a) $\Lambda_m = \Lambda_X$,

(b) $1 < c_{i_1,\dots,i_n}(m) < d$ for all $(i_1, \dots, i_n) \in \Lambda_m$,

(c) $\gcd(c_{i_1,\dots,i_n}(m), d) = 1$ for all $(i_1, \dots, i_n) \in \Lambda_m$.

Encryption Process :

(1) Choose an integer $N \in \mathbb{Z}_{>0}$ uniformly such that $Nd > 2^{\lambda} H(X)$. For an upper bound of N, see Section 3.5 below.

(2) Construct the twisted plaintext $\widetilde{m}(\underline{x}) \in \mathbb{Z}[\underline{x}]$ by putting $\Lambda_{\widetilde{m}} := \Lambda_m$ and $c_{\underline{i}}(\widetilde{m}) := c_{\underline{i}}(m)^e \pmod{Nd}$ $(0 < c_{\underline{i}}(\widetilde{m}) < Nd,\ \underline{i} \in \Lambda_{\widetilde{m}})$.

(3) Choose a random $f(\underline{x}) \in \mathbb{Z}[\underline{x}]$ uniformly such that

   (a) $\Lambda_f = \Lambda_X$,

   (b) $H(\widetilde{m}) < c_{\underline{k}}(f) < Nd$ and $\gcd\left(c_{\underline{k}}(f), d\right) = 1$, where $\underline{k}$ is the maximal element in Λ_f.

(4) Choose random polynomials $s_j(\underline{x}), r_j(\underline{x}) \in \mathbb{Z}[\underline{x}]$ uniformly with $\Gamma_{s_j} = \Gamma_X$ and $\Gamma_{r_j} = \Gamma_f$ $(j \in \{1, 2, 3\})$.

(5) Put cipher polynomials $F_1(\underline{x})$, $F_2(\underline{x})$ and $F_3(\underline{x})$ as follows:

$$F_j(\underline{x}) \quad := \quad \widetilde{m}(\underline{x}) + s_j(\underline{x}) f(\underline{x}) + r_j(\underline{x}) X(\underline{x}) \quad (1 \le j \le 3).$$

Finally, send $(F_1(\underline{x}), F_2(\underline{x}), F_3(\underline{x}), N)$.

3.4 Decryption

Decryption Process :

(1) By substituting $\underline{a}/d$, a zero of $X(\underline{x})$, into $F_j(\underline{x})$ $(j \in \{1, 2, 3\})$, we obtain

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

Compute

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

(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 Sections 3.4 and 3.5 of [Oku15].

Remark 3.4.1 In the algorithm in Sections 3.4 and 3.5 of [Oku15], we need to compute $\varphi(d)$ efficiently. From this, we should choose a prime number as d.

3.5 Parameter Size

In Section 5 of [Oku15], the sizes of public/secret keys and the ciphertext 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 [Oku15] to analyze the complexity of our attack.

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

$$2^{\frac{\lambda}{2}} \le d < 2^{\frac{\lambda}{2}+1}, \quad (\lambda+1)+(\frac{\lambda}{2}+1)w_X \le e < 2\left((\lambda+1)+(\frac{\lambda}{2}+1)w_X\right),$$

$$\frac{2^{\lceil\frac{\lambda}{n-1}\rceil}}{\varphi(d)}d \le |a_i| < \frac{2^{\lceil\frac{\lambda}{n-1}\rceil+1}}{\varphi(d)}d \ (i\in\{1,\ldots,n\}), \quad 2^{\lambda+(\frac{\lambda}{2}+1)(w_X-1)} \le N < 2^{\lambda+1+(\frac{\lambda}{2}+1)(w_X-1)}.$$

We assume that $|c_{\underline{i}}(X)| < 2^b$ for any $\underline{i}\in\Lambda_X\smallsetminus\{\underline{k},\underline{0}\}$, where $\underline{k}$ is the maximal element in Λ_X (cf. [Oku15], Section 5).

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

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

bits.

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

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

bits.

(4) The size of a ciphertext is at most

$$\frac{3}{2}\left({w_X}^2+w_X\right)(\lambda+1+(\lambda+2)w_X+\lceil\log_2 w_X\rceil)+\lambda+1+(\frac{\lambda}{2}+1)(w_X-1) \qquad (3.5.1)$$

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 3.5.1 (1) In Section 4.5 of [Oku15], it is pointed out that we should use a polynomial X satisfying $w_X\ge 5$, $n\ge 3$ 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 Section 5 of [Oku15], it is pointed out that for a public key X, we may choose $c_{\underline{i}}(X)\le 2^{10}$ ($\underline{i}\in\Lambda_X\smallsetminus\{\underline{k},\underline{0}\}$), where $\underline{k}$ is the maximal element in 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 $c_{\underline{i}}(X)$ ($\underline{i}\in\Lambda_X\smallsetminus\{\underline{k},\underline{0}\}$) to deal with a wide class of polynomials of degree increasing type. In our experiments of Section 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 which is about 10 times smaller than sizes of public keys in well-known efficient candidates of PQC [LPR10, MTSB13, TSTD13].

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 : $m(\underline{x}) = m(x, y) = 3x^3 + 3y + 2$.

Objects for Encryption :

(1) $N = 353408$ $(Nd = 1767040)$.

(2) $\widetilde{m}(\underline{x}) = \widetilde{m}(x, y) = 146243x^3 + 146243y + 131072$ $(H(\widetilde{m}) = 146243)$.

(3) $f(\underline{x}) = f(x, y) = 949843x^3 + 1324952y + 1109775$.
$(c_{\underline{k}}(f) = 949843$, $H(\widetilde{m}) = 146243 < c_{\underline{k}}(f) = 949843 < 1767040 = Nd.)$

(4) s_j and r_j $(1 \leq j \leq 3)$:

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

(5) *Cipher Polynomials* : $F_j(\underline{x}) := \widetilde{m}(\underline{x}) + s_j(\underline{x}) f(\underline{x}) + r_j(\underline{x}) X(\underline{x}) \quad (1 \leq j \leq 3)$.

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

4 Attack against DEC in Polynomial Time via Weighted LLL Reduction

In this section, we present an algorithm of our attack against DEC via the weighted LLL algorithm. We use the following notation described in Notation of Section 1: for a polynomial $h = \sum_{\underline{i} \in \Lambda_h} c_{\underline{i}}(h) x^{\underline{i}} \in \mathbb{Z}[\underline{x}]$, we define $\mathbf{h} := \left(c_{\underline{i}_1}(h), \ldots, c_{\underline{i}_{\#\Lambda_h}}(h)\right)$, where $\Lambda_h = \{\underline{i}_1, \ldots, \underline{i}_{\#\Lambda_h}\}$ is the support of h. Note that Λ_h is an ordered set (see Notation in Section 1).

Recall from Sections 3.2 and 3.3 that a public key and a ciphertext are $(d, e, X) \in \mathbb{Z}^2 \times (\mathbb{Z}[\underline{x}])$ and $(F_1, F_2, F_3, N) \in (\mathbb{Z}[\underline{x}])^3 \times \mathbb{Z}$, respectively. Let $m \in \mathbb{Z}[\underline{x}]$ be a plaintext. Note that $F_j = \widetilde{m} + s_j f + r_j X$ for $1 \leq j \leq 3$, where $\widetilde{m}$, f, s_j and r_j are the twisted plaintext and random polynomials chosen uniformly according to Section 3.3, respectively. We fix $\Lambda_X = \{\underline{i}_1, \ldots, \underline{i}_q\}$ with $\underline{i}_1 \succ \cdots \succ \underline{i}_q$, where the total order $\succ$ on Λ_X is given in Remark 3.1.2 (2). Throughout this section, put $q := \#\Lambda_X$ Let $\underline{k}$ be the maximal element in Λ_X. Note that it is sufficient for recovering the plaintext m to recover $\widetilde{m}$, and that the supports of m, $\widetilde{m}$, s_j, r_j and f $(1 \leq j \leq n)$ are the same as Λ_X. This condition on the supports of polynomials allows us to assume $\Lambda_{F_1} = \Lambda_{F_2} = \Lambda_{F_3}$.

4.1 Idea of Our Attack

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

$$F_j(\underline{x}) := \widetilde{m}(\underline{x}) + s_j(\underline{x}) f(\underline{x}) + r_j(\underline{x}) X(\underline{x}) \qquad \text{for } 1 \leq j \leq 3.$$

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

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

$$F_j(\underline{x}) - F_{j+1}(\underline{x}) = (s_j(\underline{x}) - s_{j+1}(\underline{x})) f(\underline{x}) + (r_j(\underline{x}) - r_{j+1}(\underline{x})) X(\underline{x}). \qquad (4.1.1)$$

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

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

We then have the following equalities:

$$\begin{aligned} F'_1(\underline{x}) &= s'_1(\underline{x}) f(\underline{x}) + r'_1(\underline{x}) X(\underline{x}), & (4.1.2) \\ F'_2(\underline{x}) &= s'_2(\underline{x}) f(\underline{x}) + r'_2(\underline{x}) X(\underline{x}), & (4.1.3) \\ g(\underline{x}) X(\underline{x}) &= s'_2(\underline{x}) F'_1(\underline{x}) - s'_1(\underline{x}) F'_2(\underline{x}). & (4.1.4) \end{aligned}$$

4.1.1 Step 1: Determination of s'_j for $j = 1$ and 2

Here, we describe how to determine $\mathbf{s}'_j$ for $j = 1$ and 2. (As we mentioned in Section 1, the vectors $\mathbf{s}'_j$ $(j = 1$ and $2)$ are the most important target vectors). In the equality (4.1.4), we regard the coefficients of $s'_j(\underline{x})$ and $g(\underline{x})$ as variables. We then obtain the linear system $\mathbf{u}A = \mathbf{0}$, where A is a $((2q + \#\Lambda_{X^2}) \times \#\Lambda_{X^3})$ matrix. We denote by $\mathcal{L}'_1$ the kernel lattice of A, where the kernel lattice of A is defined as the nullspace of A (see Notation in Section **??**). 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 6.0.1 in Section 6). Thus, we assume the following condition:

Assumption 4.1.1 The rank of $\mathcal{L}_1$ is equal to 3.

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

$$w'_j := \begin{cases} 1 & \left(c_{\underline{i}_j}(X) = H(X)\right), \\ 2^{\left\lfloor \log_2 \left(\frac{H(X)}{c_{\underline{i}_j}} \right) \right\rfloor} & \text{(otherwise)}, \end{cases}$$

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

Assumption 4.1.2 The $(\mathbf{s}'_1, \mathbf{s}'_2)$ is a shortest vector in $\mathcal{L}_1^{\mathbf{w}}$.

Let f_W be the isomorphism described in Section 2 from $\mathbb{R}^{2q}$ to $\mathbb{R}^{2q}$ as $\mathbb{R}$-vector spaces. From Assumption 4.1.1, the rank of $f_W(\mathcal{L}_1)$ is equal to 3. This means that we can expect the weighted LLL algorithm for the weight $\mathbf{w}$ to output a shortest vector in $\mathcal{L}_1^{\mathbf{w}}$ with high probability. Thus it is expected to find the correct $(\mathbf{s}'_1, \mathbf{s}'_2)$ via the weighted LLL algorithm for the weight $\mathbf{w}$ because of Lemma 2.1.2 and Assumption 4.1.2.

Remark 4.1.3 We may fail in determining $\mathbf{s}'_j$ $(j = 1, 2)$ if we apply the LLL algorithm with respect to the p-norm $(1 \le p \le \infty)$ to the lattice $\mathcal{L}_1$ as we will see in Section 4.3. Thus, the above assumptions and applying the weighted LLL algorithm to $\mathcal{L}_1$ are crucial for our attack.

4.1.2 Step 2: Fixing of 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 (4.1.2) and (4.1.3). 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 variables, we have the linear system. We then fix $f'(\underline{x})$ such that (4.1.2) and (4.1.3) 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 4.1.9 and Steps 3-3 and 3-4 in Section 4.2).

Remark 4.1.4 In Step 2, any solution (f', r''_1) of 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

$$\begin{aligned} F'_1 &= s'_1 f' + r''_1 X \\ &= s'_1 (f + p) + \left(r'_1 + q\right) X \\ &= \left(s'_1 f + r'_1 X\right) + \left(s'_1 p + qX\right) \\ &= F'_1 + \left(s'_1 p + qX\right). \end{aligned}$$

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 $\deg\, p \le \deg\, X$ and $\deg\, q \le \deg\, s'_1$. This fact implies that the rank of the kernel lattice in Step 2 is equal to 1 with high probability. 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 Section 3.2, we have $\gcd(X, s'_1) = 1$ with high probability.

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

Here, we describe how to recover $\widetilde{m}$. It is sufficient for recovering $\widetilde{m}(\underline{x})$ to find $\mathbf{s}_1$ (cf. Remark 4.1.9 and Steps 3-3 and 3-4 in Section 4.2). From the form of the ciphertext (see Section 3.3), we consider the following equality:

$$F_1(\underline{x}) \quad = \quad \widetilde{m}(\underline{x}) + s_1(\underline{x}) f'(\underline{x}) + r_1(\underline{x}) X(\underline{x}), \tag{4.1.5}$$

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 $F_1 = \widetilde{m} + s_1 f' + rX$ (cf. Remark 4.1.9). 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 variables, we have the linear system $\mathbf{w}C = \mathbf{c}$, where C is a $(3q \times \#\Lambda_{X^2})$ matrix and $\mathbf{c} \in \mathbb{Z}^{\#\Lambda_{X^2}}$. We denote by $\mathcal{L}_3$ the kernel lattice of C. The rank of $\mathcal{L}_3$ is equal to 3 with high probability (see Remark 6.0.1). From this, we assume the rank of $\mathcal{L}_3$ as follows:

Assumption 4.1.5 The rank of $\mathcal{L}_3$ is equal to 3.

Let $\mathbf{w}_0$ be one solution of $\mathbf{w}C = \mathbf{c}$ and $\{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3\}$ a basis of $\mathcal{L}_3$. Note that every integral solution of the system is represented as $\mathbf{w}_0 + a_1\mathbf{w}_1 + a_2\mathbf{w}_2 + a_3\mathbf{w}_3$ ($a_i \in \mathbb{Z}$, $i = 1, 2$ and 3). The 1-$\#\Lambda_X$-th entries of $\mathbf{w}_0$, $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$ correspond to the coefficients of $\widetilde{m}$. We also note that $\mathbf{w}C = \mathbf{0}$ has a solution $\mathbf{w}'$ such that its 1-$\#\Lambda_X$-th entries are equal to 0 (see Remark 4.1.8). We choose such a solution as $\mathbf{w}_3$. Assume the following condition:

Assumption 4.1.6 The vector $\mathbf{s}_1$ coincides with the vector consisting of the $(\#\Lambda_X + 1)$-$2\#\Lambda_X$-th entries of $\mathbf{w}_0 + \mathbf{w}_3 - \mathbf{z}$, where $\mathbf{z}$ is a closest vector in $\mathcal{L}'_3 := \langle \mathbf{w}_1, \mathbf{w}_2 \rangle_{\mathbb{Z}}$ to $\mathbf{w}_0 + \mathbf{w}_3$.

The rank of $\mathcal{L}'_3$ is equal to 2, and thus we can expect to find $\mathbf{s}_1$ in polynomial time by Babai's nearest plane algorithm [Bab86] for solving CVP with sufficiently high probability under Assumption 4.1.6.

Remark 4.1.7 The reason why we assume Assumption 4.1.6 is the following: From the choice of $\mathbf{s}_1$, the absolute values of the entries of $\mathbf{s}_1$ are sufficiently smaller than those of $\widetilde{\mathbf{m}}$ and $\mathbf{r}_1$. Thus we can expect that the value of $\|\mathbf{w}_0 + \mathbf{w}_3 - (a_1\mathbf{w}_1 + a_2\mathbf{w}_2)\|$ is sufficiently small if certain entries of $\mathbf{w}_0 + \mathbf{w}_3 - (a_1\mathbf{w}_1 + a_2\mathbf{w}_2)$ coincide with those of $\mathbf{s}_1$.

Remark 4.1.8 In Step 3, the linear system $\mathbf{w}C = \mathbf{0}$ has a solution $\mathbf{w}'$ such that its 1-$\#\Lambda_X$-th entries are equal to 0. Let $(\mathbf{m}', \mathbf{s}', \mathbf{r}')$ be one solution of $\mathbf{w}C = \mathbf{c}$, i.e., $F_1 = m' + s'f' + r'X$. The vector $(\mathbf{m}', \mathbf{s}', \mathbf{r}') + (\mathbf{0}, \mathbf{X}, -\mathbf{f}')$ is also a solution of $\mathbf{w}C = \mathbf{c}$. In fact, we have

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

Thus $(\mathbf{0}, \mathbf{X}, -\mathbf{f}')$ is an element of the kernel lattice of C.

Remark 4.1.9 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 4.1.4. 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

We write down an algorithm of the attack based on the idea described in Section 4.1. Recall from Section 3 that a public key and a ciphertext are $(d, e, X) \in \mathbb{Z}^2 \times (\mathbb{Z}[\underline{x}])$ and $(F_1, F_2, F_3, N) \in (\mathbb{Z}[\underline{x}])^3 \times \mathbb{Z}$, respectively. Let $m \in \mathbb{Z}[\underline{x}]$ be a plaintext. Note that $F_j = \widetilde{m} + s_j f + r_j X$, where $\widetilde{m}$, f, s_j and r_j are the twisted plaintext and random polynomials chosen uniformly according to Section 3.3, respectively. We also recall that Λ_X and w_X denote the support of X and the total degree of X, respectively (see Notation in Section 1). Let $\underline{k}$ be the maximal element in Λ_X with respect to the order described in Remark 3.1.2 (2).

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

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 $g := s'_2 r'_1 - s'_1 r'_2$. Solve the linear system $\mathbf{u}A = \mathbf{0}$ obtained by comparing the coefficients of the equality

$$s'_2 F'_1 - s'_1 F'_2 \;=\; gX, \tag{4.2.1}$$

where A is a $(2\#\Lambda_X + \#\Lambda_{X^2}) \times \#\Lambda_{X^3}$ matrix. Let $\{\mathbf{u}'_1, \mathbf{u}'_2, \mathbf{u}'_3\}$ be the basis of the kernel lattice of A.

Step 1-2. Let $\mathbf{u}_i$ be the vector consisting of the 1-$(2\#\Lambda_X)$-th entries of $\mathbf{u}'_i$ for $i = 1$, 2 and 3. Execute the weighted LLL algorithm for the weight described in Section 4.1 to the lattice $\mathcal{L}_1 := \langle \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3 \rangle$, and then obtain $\mathbf{s}'_1$ and $\mathbf{s}'_2$.

Step 2. Fixing of a candidate of f
Step 2-1. Solve the linear system $\mathbf{v}B = \mathbf{b}$ obtained by comparing the coefficients of the equalities

$$F'_1 = s'_1 f + r'_1 X, \quad F'_2 \;=\; s'_2 f + r'_2 X, \tag{4.2.2}$$

where B is a $(3\#\Lambda_X \times \#\Lambda_{X^2})$ matrix. Let $\mathbf{v}_0$ be a solution of $\mathbf{v}B = \mathbf{b}$ and $\{\mathbf{v}_1\}$ a basis of the kernel lattice $\mathcal{L}_2$ of B. Note that if $\gcd(X, s'_1) = 1$, then $\mathcal{L}_2$ is always a lattice of 1-rank (cf. Remark 4.1.4).

Step 2-2. Let $\mathbf{v}_0' := \mathbf{v}_0 - \lfloor \langle \mathbf{v}_0, \mathbf{v}_1 \rangle / \langle \mathbf{v}_1, \mathbf{v}_1 \rangle \rceil \mathbf{v}_1$ be the other solution of $\mathbf{v}B = \mathbf{b}$. Let $\mathbf{v}_0''$ be the vector consisting of the 1-($\#\Lambda_X$)-th entries of $\mathbf{v}_0'$. Construct $f'(\underline{x}) \in \mathbb{Z}[\underline{x}]$ so that $\mathbf{f}' = \mathbf{v}_0''$. Note that $\mathbf{v}_0'$ provides the polynomial closer to the correct f than $\mathbf{v}_0$ in many cases (cf. Step 2 in Section 4.3).

Step 3. Recovery of $\widetilde{m}$
Step 3-1. Solve the linear system $\mathbf{w}C = \mathbf{c}$ obtained by comparing the coefficients of the equality

$$F_1 = \widetilde{m} + s_1 f' + r_1 X, \tag{4.2.3}$$

where C is a $(3\#\Lambda_X \times \#\Lambda_{X^2})$ matrix and f' is the polynomial obtained in Step 2-2. Let $\mathbf{w}_0$ be a solution of $\mathbf{w}C = \mathbf{c}$ and $\{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3\}$ a basis of the kernel lattice $\mathcal{L}_3$ of C.

Step 3-2. Execute Babai's nearest plane algorithm to find a closest vector $\mathbf{z}$ in the lattice $\mathcal{L}_3' := \langle \mathbf{w}_1, \mathbf{w}_2 \rangle_{\mathbb{Z}}$ to $\mathbf{w}_0 + \mathbf{w}_3$. Let $\mathbf{s}_1$ be the vector consisting of the $(\#\Lambda_X + 1)$-$2\#\Lambda_X$-th entries of $\mathbf{w}_0 + \mathbf{w}_3 - \mathbf{z}$.

Step 3-3. Solve the linear system $\mathbf{x}H = \mathbf{h}$ obtained by comparing the coefficients of the equality

$$F_1 - \widetilde{m} - s_1 f' = rX, \tag{4.2.4}$$

where the coefficients of $\widetilde{m}$ and r are variables and H is a $(2\#\Lambda_X \times \#\Lambda_{X^2})$ matrix. Let $\mathbf{x}$ be a solution of $\mathbf{x}H = \mathbf{h}$. Let $\mathbf{r}'$ be the vector consisting of the entries corresponding to r of $\mathbf{x}$. 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

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

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

Remark 4.2.1 In Step 3-4 of the above algorithm, we use the fact that $c_{\underline{k}}(X)$ is divisible by d to compute an integer t (see (3.2.1) for the divisibility of $c_{\underline{k}}(X)$).

4.3 Cryptanalysis of Toy Example

We break the one-wayness of the instance in Section 3.6 of DEC. We use the same notations as in Section 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 Step 1: Determination of $s'_j = s_j - s_{j+1}$ for $j = 1$ and 2

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

$$\begin{aligned}
F'_1(x,y) &:= F_1(x,y) - F_2(x,y) \\
&= 11140319x^6 - 129039168x^3y + 52374932282x^3 - 199195126y^2 + 72078461753y + 115343333092, \\
F'_2(x,y) &:= F_2(x,y) - F_3(x,y) \\
&= 3327020x^6 - 107361990x^3y - 20505232066x^3 - 311567430y^2 + 6054708965y - 130662866703.
\end{aligned}$$

We put

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

where $c_i^{(j)}$'s and $c_i^{(g)}$'s are variables. By comparing the coefficient of $\underline{x}^{\underline{i}}$ for each $\underline{i} \in \Lambda_{X^3}$ in the equation (4.1.4), we have the linear system $\mathbf{u}A' = \mathbf{0}$, where A' is a (9×9) matrix.
The rank of the kernel lattice $\mathcal{L}'_1$ of A' is equal to 3. Compute a basis $\{\mathbf{u}'_1, \mathbf{u}'_2, \mathbf{u}'_3\}$ of $\mathcal{L}'_1$, and let $\mathbf{u}_j$ be the vector of the 1-6th entries of $\mathbf{u}'_j$ for $j = 1$, 2 and 3. We then have

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

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

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

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

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

Thus $\mathbf{a}_i$ does not coincide with both of $(\mathbf{s}'_1, \mathbf{s}'_2)$ and $-(\mathbf{s}'_1, \mathbf{s}'_2)$ for any $1 \leq i \leq 3$. Note that 1-2nd and 4-5th entries of the correct $(\mathbf{s}'_1, \mathbf{s}'_2)$ 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 Section 3 of [Oku15] and Section 4.2 in this paper. On the other hand, the absolute values of all entries of $\mathbf{a}_i$ have almost the same sizes for $1 \leq i \leq 3$. Moreover, it is easy to see $\|(\mathbf{s}'_1, \mathbf{s}'_2)\|_p > \max\{\|\mathbf{a}_1\|_p, \|\mathbf{a}_2\|_p, \|\mathbf{a}_3\|_p\}$ for any $1 \leq p \leq \infty$, where $\|\cdot\|_p$ denotes the p-norm. For example, we have $\|(\mathbf{s}'_1, \mathbf{s}'_2)\|_2 \approx 48383.47 > \max\{\|\mathbf{a}_1\|_2, \|\mathbf{a}_2\|_2, \|\mathbf{a}_3\|_2\} \approx 21418.08$. This means that our target vector $(\mathbf{s}'_1, \mathbf{s}'_2)$ is not shortest in $\mathcal{L}_1$ of 3-rank with respect to the p-norm for any $1 \leq p \leq \infty$. Thus, it seems that the LLL algorithm with respect to well-known norms such as the p-norms $(1 \leq p \leq \infty)$ does not work well for finding $(\mathbf{s}'_1, \mathbf{s}'_2)$.

To obtain $(\mathbf{s}'_1, \mathbf{s}'_2)$, we apply the weighted LLL algorithm for the weight $\mathbf{w}$ described below to $\mathcal{L}_1$ since the above situation is good for the weighted LLL algorithm (see Section 4 in [FGR13]). Recall that $\mathbf{X} = (125, 675, -110438)$. 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

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

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

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

Note that $\mathbf{b}_1$ coincides with the target vector $(\mathbf{s}_1', \mathbf{s}_2')$.

4.3.2 Step 2: Fixing of A Candidate of f

Here, we fix a candidate of f. We set

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

where $c_i^{(f)}$'s and $c_i^{(j)}$'s are variables. By substituting s_1' and s_2' obtained in Step 1 into the equalities (4.1.2) and (4.1.3), and by comparing the coefficient of $\underline{x}^{\underline{i}}$ for each $\underline{i} \in \Lambda_{X^2}$, we have the linear system $\mathbf{v}B = \mathbf{b}$, where B is a (9×6) matrix. The rank of the kernel lattice $\mathcal{L}_2$ of B is equal to 1. We obtain a solution $\mathbf{v}_0$ of $\mathbf{v}B = \mathbf{b}$ and a basis $\{\mathbf{v}_1\}$ of $\mathcal{L}_2$ as follows:

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

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

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

We set

$$f'(x,y) \;:=\; 950468x^3 + 1328327y + 557585. \tag{4.3.1}$$

Note that the polynomial f' obtained from $\mathbf{v}_0''$ is closer to the correct f than the one obtained from $\mathbf{v}_0$. We also note that it is possible to proceed to the next step even if f' does not coincide with f (cf. Remark 4.1.9).

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

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

$$\widetilde{m}(x,y) \;:=\; c_1 x^3 + c_2 y + c_3, \tag{4.3.2}$$

$$s_1(x,y) \;:=\; c_4 x^3 + c_5 y + c_6, \tag{4.3.3}$$

$$r_1(x,y) \;:=\; c_7 x^3 + c_8 y + c_9, \tag{4.3.4}$$

where c_i's are variables. By substituting f' obtained in Step 2 into the equalities (4.1.5), and by comparing the coefficient of $\underline{x}^{\underline{i}}$ for each $\underline{i} \in \Lambda_{X^2}$, we have the linear system $\mathbf{w}C = \mathbf{c}$, where C is a (9×6) matrix. The rank of the kernel lattice $\mathcal{L}_3$ of C is equal to 3. We fix a solution $\mathbf{w}_0$ of the system and a basis $\{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3\}$ of $\mathcal{L}_3$ as follows:

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

We find a vector $\mathbf{z}$ of the lattice $\langle \mathbf{w}_1, \mathbf{w}_2 \rangle_{\mathbb{Z}}$ close to $\mathbf{w}_0 + \mathbf{w}_3$ by applying Babai's 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 $\mathbf{z}$ and $\mathbf{w}_0 + \mathbf{w}_3 - \mathbf{z}$, respectively. The vector consisting of the 4-6th entries of $\mathbf{w}_0 + \mathbf{w}_3 - \mathbf{z}$ is equal to the correct $\mathbf{s}_1$.

Next, we compute $r(x, y)$ satisfying $F_1(x, y) - \widetilde{m}(x, y) - s_1'(x, y) = r(x, y) X(x, y)$. 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 4.1.9 and Step 3-4 in Section 4.2). We set

$$\begin{aligned} r(x, y) &:= c_1 x^3 + c_2 y + c_3, && (4.3.5) \\ \widetilde{m}(x, y) &:= c_4 x^3 + c_5 y + c_6, && (4.3.6) \end{aligned}$$

where c_i's are variables. In the equality $F_1(x, y) - s_1(x, y) f'(x, y) = \widetilde{m}(x, y) + r(x, y) X(x, y)$, by comparing the coefficient of $\underline{x}^{\underline{i}}$ for each $\underline{i} \in \Lambda_{X^2}$, we have the linear system $\mathbf{x}H = \mathbf{h}$, where H is a (6×6) matrix. The rank of the kernel lattice $\mathcal{L}_4$ of H is equal to 1. We fix a solution $\mathbf{x}_0$ of $\mathbf{x}H = \mathbf{h}$ and a basis $\{\mathbf{x}_1\}$ of $\mathcal{L}_4$ as follows:

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

We set

$$r'(x, y) := 2591380 x^3 + 4015684 y + 1. \qquad (4.3.7)$$

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

$$\begin{aligned} e' &:= e^{-1} \pmod{\varphi(d)} \\ &= 1, \\ H_1(x, y) &:= F_1(x, y) - s_1(x, y) f'(x, y) - r'(x, y) X(x, y) \\ &= 226710368 x^3 + 1223592518 y - 200170179622, \\ \mu &:= c_{\underline{k}}(H_1) \\ &= -200170179622, \\ c_{\underline{k}}(m') &:= \mu^{e'} \pmod{d} \quad \left(0 < c_{\underline{k}}(m') < d\right) \\ &= 3, \\ c_{\underline{k}}(\widetilde{m}) &= \left(c_{\underline{k}}(m')\right)^e \pmod{Nd} \quad \left(0 < c_{\underline{k}}(\widetilde{m}) < Nd\right) \\ &= 146243, \end{aligned}$$

$$
\begin{aligned}
t &= \left(\mu - c_{\underline{k}}\left(\widetilde{m}\right)\right)/c_{\underline{k}}\left(X\right)\\
&= 1812513,\\
\widetilde{m}\left(x,y\right) &= F_1\left(x,y\right) - s_1\left(x,y\right) f'\left(x,y\right) - \left(r'\left(x,y\right) + t\right) X\left(x,y\right)\\
&= 146243x^3 + 146243y + 131072.
\end{aligned}
$$

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

5 Complexity Analysis

In this section, we investigate the complexity of the algorithm in Section 4.2. We analyze our attack according to the parameter sizes in Section 3.5 (cf. Section 5 in [Oku15]). 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 with respect to the parameters w and λ. The parameters d and e are $O\left(2^\lambda\right)$ and $O\left(w\lambda\right)$, respectively. Note that the size of each coefficient of F_j is $O\left(w\lambda\right)$ bits for $j = 1$, 2 and 3 (see Section 3.5 for the representation of the parameters by w and λ).

Remark 5.0.1 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 $O\left(\min\{q_F, q_G\}\right)$ 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 $O\left(\left(\max\{q_F, q_G\}\right)^2\right)$ 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 $O\left(w\lambda\right)$ 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

$$
O\left(w^2\left(w\lambda\right)\right) \quad = \quad O\left(w^3\lambda\right). \tag{5.0.1}
$$

We do such computations in (4.2.1)-(4.2.4). The arithmetic complexity of (4.2.1)-(4.2.4) is $O(w^4)$ and thus the bit complexity is

$$
O\left(w^4\left(w\lambda\right)^2\right) \quad = \quad O\left(w^6\lambda^2\right) \tag{5.0.2}
$$

since the sizes of the coefficients of the polynomials appearing in (4.2.1)-(4.2.4) are $O\left(w\lambda\right)$ bits. Note that we regard the coefficients of certain polynomials as variables. (For example, in (4.2.1), we regard the coefficients of ${s_1}'$, ${s_2}'$ and g as variables.) 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 variables. 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 $O\left(w\lambda\right)$ bits. Thus the bit complexity of computing H_1 is

$$
O\left(w^2\left(w\lambda\right)^2\right) \quad = \quad O\left(w^4\lambda^2\right). \tag{5.0.3}
$$

Remark 5.0.2 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 Hermite Normal Form (HNF) of the augmented matrix of the system. According to Chapter 2 in [Gal12], we assume that the computation of the HNF of an $n \times m$ matrix $M = (M_{i,j})_{i,j}$ requires $O(nm^4(\log(\|M\|_\infty))^2)$ bit operations, where $\|M\|_\infty := \max_{i,j}\{|M_{i,j}|\}$. 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 5.0.3 Third, we discuss the size of the norm of a vector with integer entries. Let $\mathbf{a} = (a_1, \ldots, a_k) \in \mathbb{Z}^k$ be a vector with $|a_i| \le 2^l$ for $1 \le i \le k$. Since $\|\mathbf{a}\| \le \sqrt{k2^{2l}}$, the size of $\|\mathbf{a}\|$ 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 $\|\mathbf{a}\|^2$ is $O(\log(k) + l)$ bits.

5.1 The Complexity of Step 1

Step 1-1. We estimate the bit complexity for solving the linear system $\mathbf{u}A = \mathbf{0}$ with at most $2w + w^2$ variables and w^3 equations. Since this linear system is homogeneous, the arithmetic complexity in $\mathbb{Z}$ of solving the linear system is $O\left(w^6\right)$ (see Remark 5.0.2). The size of each entry of A is $O(w\lambda)$ bits, and thus Step 1-1 requires

$$O\left(w^8\lambda^2\right) \tag{5.1.1}$$

bit operations. In addition, we note that the sizes of the entries of $\mathbf{u}_1', \mathbf{u}_2'$ and $\mathbf{u}_3'$, that are basis vectors of the kernel lattice $\mathcal{L}_1'$ of A, are $O(w\lambda)$ bits from Remark 5.0.2.

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

$$O\left(w \cdot (w\lambda)^2\right) = O\left(w^3\lambda^2\right) \tag{5.1.2}$$

bit operations. We note that the size of each entry of UW is $O(w\lambda)$ bits. After the multiplying, we execute the LLL algorithm to the $2w$-dimensional lattice $f_W(\mathcal{L}_1)$ of 3-rank with the basis matrix UW. According to [LLL82], the computation of the LLL algorithm requires $O\left(3^5(2w)\left(\log\left(2w \cdot 2^{2w\lambda}\right)\right)^3\right)$ bit operations in this case because the norms of the row vectors of UW are $O\left(\sqrt{2w \cdot 2^{2w\lambda}}\right)$. Thus the LLL algorithm of this step runs in

$$O\left(w^4\lambda^3\right) \tag{5.1.3}$$

bit operations. Any entry of the vectors of the LLL reduced basis is $O\left(\sqrt{3\left(w \cdot 2^{2w\lambda}\right)}\right)$ because the rank of $f_W(\mathcal{L}_1)$ is equal to 3, and because $\|\mathbf{u}_iW\|^2 = O\left(w \cdot 2^{2w\lambda}\right)$ for $i = 1$, 2 and 3, and row vectors

$\mathbf{u}_i$ of U. Thus the size of any entry of the basis matrix is $O(w\lambda)$ bits. We multiple the diagonal matrix W^{-1} by the LLL reduced basis matrix. The arithmetic complexity of the multiplying is $3 \cdot 2w = O(w)$. Thus the multiplying runs in

$$O\left(w\,(w\lambda)^2\right) \quad = \quad O\left(w^3\lambda^2\right) \tag{5.1.4}$$

bit operations.

5.2 The Complexity of Step 2

Step 2-1. In this step, we solve the linear system $\mathbf{v}B = \mathbf{b}$ with $3w$ variables and at most w^2 equations. In a similar way to Step 1-1, the bit complexity of this step can be estimated as

$$O\left(w^{11}\lambda^2\right). \tag{5.2.1}$$

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 $\mathbf{v}_0$ and $\mathbf{v}_1$ are $O\left(\sqrt{(3w \cdot 2^{2w\lambda})}\right)$.

Step 2-2. In this step, we compute $\mathbf{v}_0' := \mathbf{v}_0 - \lfloor\langle\mathbf{v}_0, \mathbf{v}_1\rangle/\langle\mathbf{v}_1, \mathbf{v}_1\rangle\rceil\mathbf{v}_1$. This computation requires $O\left(2^4\,(3w)\left(\log\left(3w \cdot 2^{2w\lambda}\right)\right)^2\right)$ bit operations according to Chapter 17 in [Gal12]. Hence Step 2-2 requires

$$O\left(w^3\lambda^2\right) \tag{5.2.2}$$

bit operations.

5.3 The Complexity of Step 3 and The Total Complexity of Our Attack

Step 3-1. We solve the linear system $\mathbf{w}C = \mathbf{c}$ with $3w$ variables and at most w^2 equations. In a similar way to Steps 1-1 and 2-1, the computation requires

$$O\left(w^{11}\lambda^2\right) \tag{5.3.1}$$

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

Step 3-2. We execute Babai's nearest plane algorithm to the $3w$-dimensional lattice $\mathcal{L}_3' := \langle\mathbf{w}_1, \mathbf{w}_2\rangle$ of 2-rank and the vector $\mathbf{w}_0 + \mathbf{w}_3$. Before executing Babai's nearest plane algorithm, we execute the LLL algorithm to $\mathcal{L}_3{}'$. Since $\mathcal{L}_3{}'$ has 2-rank and $3w$-dimension, the LLL algorithm requires $O\left(2^5\,(3w)\left(\log\left(3w \cdot 2^{2w\lambda}\right)\right)^3\right)$ bit operations. Thus the LLL algorithm 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 $O\left(\sqrt{2\,(3w \cdot 2^{2w\lambda})}\right)$ (cf. Chapter 17 in [Gal12]). Thus Babai's nearest plane algorithm requires $O\left(2^5\left(\log\left(\sqrt{2\,(3w \cdot 2^{2w\lambda})}\right)\right)^2\right)$ bit operations. From this, the bit complexity of Babai's nearest plane algorithm is $O\left(w^2\lambda^2\right)$ in this case. Hence Step 3-2 runs in

$$O\left(w^4\lambda^3\right) \tag{5.3.2}$$

bit operations.

Step 3-3. We solve the linear system $\mathbf{x}H = \mathbf{h}$ with $2w$ variables and at most w^2 equations. The size of any entry of H and $\mathbf{h}$ is $O\left(w\lambda\right)$ bits. Hence Step 3-3 runs in

$$O\left(w^{16}\lambda^2\right) \tag{5.3.3}$$

bit operations.

Step 3-4. At the beginning of this step, we compute $e' := e^{-1} \bmod \varphi\left(d\right)$ by using the extended Euclid's algorithm. According to Remark 3.5 in [Oku15], the integer d should be chosen so that one can compute $\varphi\left(d\right)$ efficiently because the computation is needed in the decryption process (see [Oku15], Section 3.4). In Remark 3.5 of [Oku15], 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 $\varphi\left(d\right) = d - 1$.

Next, we compute $c_{\underline{k}}\left(m'\right) := \mu^{e'} \pmod{d} \left(0 < c_{\underline{k}}\left(m'\right) < d\right)$, where $e' := e^{-1} \pmod{\varphi(d)}$ and μ is a certain coefficient of $H_1\left(\underline{x}\right)$ (cf. Step 3-4 in Section 4.2). Recall that the bit sizes of e', μ and d are $O\left(\lambda\right)$, $O\left(w\lambda\right)$ and $O\left(\lambda\right)$, 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 $c_{\underline{k}}\left(\widetilde{m}\right) := \left(c_{\underline{k}}\left(m'\right)\right)^e \pmod{Nd} \left(0 < c_{\underline{k}}\left(\widetilde{m}\right) < Nd\right)$. Note that the sizes of $c_{\underline{k}}\left(m'\right)$, e and Nd are $O\left(\lambda\right)$, $O\left(\log\left(w\lambda\right)\right)$ and $O\left(w\lambda\right)$ bits, respectively. Thus, the square-and-multiply algorithm requires $O\left(\left(w\lambda\right)^2 \log\left(w\lambda\right)\right)$ bit operations to compute $c_{\underline{k}}\left(\widetilde{m}\right)$. As a consequence, those modular exponential arithmetic can be performed in $O\left(\lambda^3 + w^2\lambda^2 \log\left(w\lambda\right)\right)$ bit operations. Finally, the computation of $t := \left(\mu - c_{\underline{k}}\left(\widetilde{m}\right)\right)/c_{\underline{k}}\left(X\right)$ runs in $O\left(w^2\lambda^2\right)$ bit operations.
The total bit complexity of Step 3-4 is

$$O\left(\lambda^3 + w^2\lambda^2\log\left(w\lambda\right)\right). \tag{5.3.4}$$

Putting all the steps together, namely considering (5.0.1)-(5.3.4), we can determine the complexity of our attack.

Theorem 5.3.1 The total bit complexity of the attack in Section 4.2 is

$$O\left(w^{16}\lambda^2\right) + O\left(w^6\lambda^2\right) + O\left(w^4\lambda^3\right).$$

Consequently, our attack performs in polynomial time for all the parameters λ and w_X.

Remark 5.3.2 The estimated complexity in Theorem 5.3.1 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) according to Sections 3.2 and 3.3, and measured their sizes. As a result, their average size is about 10,086,237 bits.

6 Experimental Results on Our Attack

We show experimental results[1] on our attack described in Section 4 against the DEC scheme for $n = 4$, i.e., the number of variables of a public key X is equal to 4. We conducted experiments for parameters recommended in Remark 3.5.1 which can let the DEC have 128 bit security.

Experimental Procedure
For given parameters w_X, $\#\Lambda_X$ and b, we repeat the following procedure 100 times:

1. Make a secret key and a public key according to Section 3.2.
2. By using the public key constructed above, make a ciphertext according to Section 3.3.
3. Execute **Algorithm of Proposed Attack** in Section 4.2 for the above public key and the ciphertext.

In the above second step, we construct a public key X such that $2^{b-1} \leq |c_{\underline{i}}(X)| < 2^b$ for all $\underline{i} \in \Lambda_X \setminus \{\underline{k}, \underline{0}\}$, where $\underline{k}$ is the maximal element in Λ_X with respect to the order described in Remark 3.1.2 (2). In order to show the effect of the weighted LLL algorithm on our cryptanalysis, we also execute another version of **Algorithm of Proposed Attack**. (The other version of **Algorithm of Proposed Attack** here means that the usual LLL algorithm is adopted in Step 1-2 of **Algorithm of Proposed Attack** instead of the weighted LLL algorithm.) We count the number of successes and time if $\widetilde{m}$ or $-\widetilde{m}$ are recovered.

Table 1 shows our experimental results. In Step 1 of Table 1, we show the number of successes only if we succeed in recovering the target vector $(\mathbf{s}'_1, \mathbf{s}'_2)$ or $-(\mathbf{s}'_1, \mathbf{s}'_2)$ (see Step 1 of **Algorithm of Proposed Attack** in Section 4.2). In Step 3 of Table 1, we show the number of successes only if a twisted plaintext $\widetilde{m}$ or $-\widetilde{m}$ is recovered.

From Step 1 in Table 1, we see that the weighted LLL algorithm found the target vector in Step 1 of our attack with probability being about from 70 to 90%, while the usual LLL algorithm could not find the target vector. From Step 3 in Table 1, we see that our attack with the weighted LLL algorithm could recover $\widetilde{m}$ or $-\widetilde{m}$ with probability being about from 20 to 40%, while another attack with the usual LLL algorithm could not recover $\widetilde{m}$ or $-\widetilde{m}$ at all. Thus we consider that using the weighted LLL algorithm plays a central role of our attack, and that the success probability of Step 3, that is the success probability of our attack, with the weighted LLL algorithm is sufficiently high for practical cryptanalysis.

From the viewpoint of the efficiency of the key generation, encryption and decryption of DEC, the parameters in Table 1 are practical (see also Tables 4, 5 and 6 in Section 6 of [Oku15]). These experimental results suggest that our attack with the weighted LLL algorithm can break the one-wayness of DEC efficiently for practical parameters with sufficiently high probability.

[1]We use a standard note PC with 2.60GHz CPU (Intel Corei5), 16GB memory and Mac OS X 64bit. We implemented the attack in Magma V2.21-3 [BCP97].

Table 1: Experimental results on **Algorithm of Proposed Attack** in Section 4.2 against DEC with four variables of 128 bit security. We did experiments according to **Experimental Procedure** described in the beginning of Section 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" means the average of time for performing our attack. (We show the timing data in successful cases.)

Recommended parameters for DEC (Section 3.5)			Experimental results					
			Number of successes of **Algorithm of Proposed Attack** / 100					
			Method for lattice reduction in Step 1					
Total degree of a public key X	Number of monomials of X	Value of b	Usual LLL			Weighted LLL		
			Step 1	Step 3	Ave. Time	Step 1	Step 3	Ave. Time
10	3	10	0	0	-	71	23	0.02s
10	4	10	0	0	-	81	33	0.03s
10	5	10	0	0	-	86	29	0.04s
10	6	10	0	0	-	86	35	0.06s
10	7	10	0	0	-	85	29	0.07s
10	8	10	0	0	-	92	33	0.09s
10	9	10	0	0	-	88	41	0.21s
10	10	10	0	0	-	91	32	0.25s
10	3	50	0	0	-	68	21	0.02s
10	4	50	0	0	-	82	39	0.03s
10	5	50	0	0	-	77	30	0.04s
10	6	50	0	0	-	83	33	0.07s
10	7	50	0	0	-	88	41	0.08s
10	8	50	0	0	-	93	32	0.10s
10	9	50	0	0	-	92	36	0.21s
10	10	50	0	0	-	91	34	0.28s
10	3	100	0	0	-	75	29	0.02s
10	4	100	0	0	-	78	26	0.03s
10	5	100	0	0	-	80	36	0.04s
10	6	100	0	0	-	83	31	0.07s
10	7	100	0	0	-	82	34	0.08s
10	8	100	0	0	-	95	40	0.11s
10	9	100	0	0	-	87	36	0.20s
10	10	100	0	0	-	91	38	0.27s

Remark 6.0.1 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 algorithm 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 with respect to the norm. Thus the most important vector for our attack (the target vector in Step 1) is found by the weighted LLL algorithm with high probability.

7 Conclusion

In this paper, we proposed an attack against the one-wayness of the public key cryptosystem based on Diophantine equations of degree increasing type (DEC), which has been expected to have a strongly desired feature of being post-quantum cryptosystems (PQC) having public keys of small sizes, e.g., about 1,200 bits with 128 bit security. Diophantine equations of degree increasing type is proved to be unsolvable in general. Thus solving such Diophantine equations is expected to be the most difficult problem among the other problems which have been used to construct the candidates of PQC. However, it is showed in this paper that the security of DEC does not depend on the difficulty of solving such Diophantine equations and DEC is not secure. More precisely, we show that the one-wayness of DEC can be transformed to the problem of finding certain relatively shorter vectors in lattices of low ranks obtained by linearization techniques. Our most important target vector takes a special form: it is not necessarily shortest in some lattice of low rank but only some entries are relatively small. The usual LLL algorithm with respect to well-known norms such as the p-norms ($1 \leq p \leq \infty$) does not seem to work well for finding such vectors in our attack.

The most technical point of our attack is to change the norm in the LLL algorithm from the Euclidean norm to a weighted norm which is not widely used in cryptography yet. Our heuristic analysis suggests that the most important target vector becomes a (nearly) shortest vector with respect to a weighted norm for some weight chosen appropriately. Moreover, the most important target vector is a vector in a lattice of 3-rank in many cases. Therefore, the weighted LLL algorithm, which is the LLL algorithm with respect to the weighted norm, is applied to our attack. From experimental results and complexity analysis on our attack, we proved that by choosing an appropriate weight, our attack with the weighted LLL algorithm can break the one-wayness of DEC with sufficiently high probability in polynomial time for all the practical parameters under some assumptions.

Our experimental results also suggest that the weighted LLL algorithm is effective in finding vectors with special properties: they are not shortest, but the bit length of their entries are almost known and relatively small. Thus the weighted LLL algorithm can be effective in analyzing the security of cryptosystems whose securities are reduced to finding such vectors.

References

[AG04] K. Akiyama, Y. Goto, *An Algebraic Surface Public-key Cryptosystem*, IEICE Technical Report, **104** (421), pp. 13–20, (2004).

[AG06] K. Akiyama, Y. Goto, *A Public-key Cryptosystem using Algebraic Surfaces*, In: Proceedings of PQCrypto., pp. 119–138, (2006), available at `http://postquantum.cr.yp.to/`.

[AG08] K. Akiyama, Y. Goto, *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).

[AGM09] K. Akiyama, Y. Goto, H. Miyake, *An Algebraic Surface Cryptosystem*, In: Proceedings of PKC'09, Lecture Notes in Computer Science, **5443**, pp. 425–442, Springer, Berlin Heidelberg, (2009).

[Bab86] L. Babai, *On Lovász' lattice reduction and the nearest lattice point problem*, Combinatorica, **6** (1), pp. 1–13, (1986), (Preliminary version in STACS 1985).

[BHHKP14] A. Bérczes, L. Hajdu, N. Hirata-Kohno, T. Kovács, A. Pethö, *A key exchange protocol based on Diophantine equations and S-integers*, JSIAM Letters, **6** (0), pp. 85–88, (2014).

[BBD08] D. J. Bernstein, J. Buchmann, E. Dahmen (Eds.), *Post-Quantum Cryptography*, Springer-Verlag, Berlin Heidelberg, (2009).

[BCP97] W. Bosma, J. Cannon, C. Playoust, *The Magma algebra system. I. The user language*, Journal of Symbolic Computation, **24** (3-4), pp. 235–265, (1997).

[Cop97] D. Coppersmith, *Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities*, Journal of Cryptology, **10** (4), pp. 233–260, Springer-Verlag, (1997).

[Cus95] T. W. Cusick, *Cryptoanalysis of a public key system based on diophantine equations*, Information Processing Letters, **56** (2), pp. 73–75, (1995).

[DGS06] J. Ding, J. E. Gower, D. S. Schmidt, *Multivariate Public Key Cryptosystems*, Advances in Information Security, **25**, Springer, US, (2006).

[DMR76] M. Davis, Y. Matijasevič, J. Robinson, *Hilbert's tenth problem, Diophantine equations: positive aspects of a negative solution*, Mathematical Developments Arising from Hilbert Problems, pp. 323–378, American Mathematical Society, Providence, RI., (1976).

[Eis07] K. Eisenträger, *Hilbert's Tenth Problem for function fields of varieties over number fields and p-adic fields*, Journal of Algebra, **310** (2), pp. 775–792, (2007).

[FGR13] J. -C. Faugère, C. Goyet, G. Renault, *Attacking (EC)DSA Given Only an Implicit Hint*, In: Proceedings of SAC 2012, Lecture Notes in Computer Science, **7707**, pp. 252–274, Springer, Berlin Heidelberg, (2013).

[FS10] J. -C. Faugère, P. -J. Spaenlehauer, *Algebraic Cryptanalysis of the PKC'2009 Algebraic Surface Cryptosystem*, In: Proceedings of PKC'10, Lecture Notes in Computer Science, **6056**, pp. 35–52, Springer, Berlin Heidelberg, (2010).

[Gal12] S. D. Galbraith, *Mathematics of Public Key Cryptography*, Cambridge University Press, (2012).

[HP13] N. Hirata-Kohno, A. Pethö, *On a key exchange protocol based on Diophantine equations*, Infocommunications Journal, **5** (3), pp. 17–21, Scientific Association for Infocommunications (HTE), (2013).

[Iwa08] M. Iwami, *A Reduction Attack on Algebraic Surface Public-Key Cryptosystems*, Lecture Notes in Computer Science, **5081**, pp. 323–332, Springer, Berlin Heidelberg, (2008).

[LLL82] A. K. Lenstra, H. W. Lenstra, L. Lovász, *Factoring polynomials with rational coefficients*, Mathematische Annalen, **261** (4), pp. 515–534, Springer-Verlag, (1982).

[LCL95] C. H. Lin, C. C. Chang, R. C. T. Lee, *A new public-key cipher system based upon the diophantine equations*, IEEE Transactions on Computers, **44** (1), pp. 13–19, IEEE Computer Society Washington, DC, USA, (1995).

[LPR10] V. Lyubashevsky, C. Peikert, O. Regev, *On ideal lattices and learning with errors over rings*, In: Proceedings of EUROCRYPT 2010, Lecture Notes in Computer Science, **6100**, pp. 1–23, (2010).

[MTSB13] R. Misoczki, J. P. Tillich, N. Sendrier, P. S. L. M. Barreto, *MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes*, Information Theory Proceedings (ISIT), IEEE International Symposium on Information Theory, (2013).

[NIST] A draft of the report on post-quantum cryptography NISTIR 8105, available at `http://csrc.nist.gov/publications/drafts/nistir-8105/nistir_8105_draft.pdf`.

[NSA] The web page of National Security Agency: `https://www.nsa.gov/ia/programs/suiteb_cryptography/`.

[Oku15] S. Okumura, *A public key cryptosystem based on diophantine equations of degree increasing type*, Pacific Journal of Mathematics for Industry, **7** (4), pp. 33–45, Springer, Berlin Heidelberg, (2015).

[Phe91] T. Pheidas, *Hilbert's Tenth Problem for fields of rational functions over finite fields*, Inventiones mathematicae, **103** (1), pp. 1–8, Springer-Verlag, (1991).

[TSTD13] C. Tao, A. Diene, S. Tang, J. Ding, *Simple Matrix Scheme for Encryption*, In: Proceeding of PQCrypto 2013, Lecture Notes in Computer Science, **7932**, pp. 231–242, (2013).

[UT07] S. Uchiyama, H. Tokunaga, *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).

[Vid94] C. R. Videla, *Hilbert's Tenth Problem for Rational Function Fields in Characteristic 2*, Proceedings of the American Mathematical Society, **120** (1), pp. 249–253, American Mathematical Society, (1994).

[Vol07] F. Voloch, *Breaking the Akiyama-Goto cryptosystem*, Contemporary mathematics, Arithmetic, Geometry, Cryptography and Coding Theory, **487**, pp. 113–118, American Mathematical Society, Providence, RI., (2007).

[Yos11] H. Yosh, *The Key Exchange Cryptosystem Used with Higher Order Diophantine equations*, International Journal of Network Security & Its Applications Journal, **3** (2), pp. 43–50, (2011).