Cryptanalysis of a Public Key Cryptosystem Based on Diophantine Equations via Weighted LLL Reduction (Short Paper)

Jintai Ding[1], Momonari Kudo[2(✉)], Shinya Okumura[3(✉)], Tsuyoshi Takagi[4], and Chengdong Tao[5]

[1] University of Cincinnati, Cincinnati, USA
[2] Graduate School of Mathematics, Kyushu University, Fukuoka, Japan
m-kudo@math.kyushu-u.ac.jp
[3] Institute of Systems, Information Technologies and Nanotechnologies, Fukuoka, Japan
s-okumura@isit.or.jp
[4] Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan
[5] South China University of Technology, Guangzhou, China

Abstract. Okumura proposed a candidate of post-quantum cryptosystem based on Diophantine equations of degree increasing type (DEC). Sizes of public keys in DEC are small, e.g., 1,200 bits for 128 bit security, and it is a strongly desired property in post-quantum erea.

In this paper, we propose a polynomial time attack against DEC. We show that the one-wayness of DEC is reduced to finding special (relatively) short vectors in some lattices. The usual LLL algorithm does not work well for finding the most important target vector in our attack. The most technical point of our method is to heuristically find a special norm called a weighted norm to find the most important target vector. We call this method *"weighted LLL algorithm"* in this paper. Our experimental results suggest that our attack can break the one-wayness of DEC for 128 bit security with sufficiently high probability.

Keywords: Post-quantum cryptosystem · Weighted LLL reduction · Diophantine equation

1 Introduction

Post-quantum cryptography is now receiving a great amount of attention. Many candidates of post-quantum cryptosystems (PQC) have been proposed up to the

The full version [12] is available from IACR Cryptology ePrint Archive, 2015/1229.
S. Okumura—Research conducted while at Institute of Mathematics for Industry, Kyushu University.

present, e.g., lattice-based cryptosystems, code-based cryptosystems and multivariate public key cryptosystems (see [7,11]). However, these PQC require public keys of large sizes, and thus it is 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 [13]) and has been expected to be used to construct PQC. (The Diophantine problem here means that given multivariate polynomials with integer coefficients, find common rational zeros of them.) In fact, a public key cryptosystem [20] and key exchange protocols [6,17,29] based on the difficulty of the Diophantine problem have been already proposed. However, the one-wayness of the cryptosystem [20] is broken [10], and the protocols [6,17,29] are said to be impractical [17, Proposition 2].

The Diophantine problem is generalized to problems over arbitrary rings. The Algebraic Surface Cryptosystem (ASC) [4] is based on the difficulty of the Diophantine problem over global function fields which is proved to be unsolvable in general [24,27]. In [4], 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 [15].

Okumura [23] proposed a public key cryptosystem based on the difficulty of solving Diophantine equations of degree increasing type over $\mathbb{Z}$ (see Sect. 3 in this paper for the definition of a polynomial of degree increasing type). We refer to the cryptosystem as DEC for short. In [23, Remark 3.2], 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. In DEC, a polynomial $X \in \mathbb{Z}[x_1, \ldots, x_n]$ of degree increasing type is used as a public key, and a vector $(a_1, \ldots, a_n) \in \mathbb{Z}^n$ satisfying $X(\frac{a_1}{d}, \ldots, \frac{a_n}{d}) = 0$ for some $d \in \mathbb{Z}$ is used as a secret key (see Sect. 3.2). The main ideas to avoid the analogues of all attacks [15,18,26,28] against ASC (and the previous versions [1–3] 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 [23, Sect. 4], 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.

Another advantage of DEC is that sizes of public keys are small, e.g., about 1,200 bits with 128 bit security (see [12, Remark 3.5.1 (3)]), as desired in post-quantum cryptography. Note that well-known efficient candidates of PQC [21,22,25] 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.

Our Contribution

In this paper, we propose an attack against DEC. We show that the one-wayness of DEC is transformed to a problem of finding special relatively short vectors in lattices obtained from a public key and a ciphertext. Using three polynomials as a ciphertext allows us to construct linear systems. One can recover a plaintext by finding appropriate solutions of the linear systems, and thus this point is one of the weakness of DEC. Our attack can be divided roughly into three steps, and our experimental results in Sect. 5 show that the key point of our attack is whether a first step of our attack 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 such as the LLL algorithm [19]. However, in [12, Sect. 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.

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 (see Sect. 2 for weighted norms). Note that using other well-known norms in the LLL algorithm does not seem to be effective in finding the target vector (see [12, Sect. 4.3]). 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 [9] (see also [16, Chap. 19]) and in [14]. In particular, we consider 2-power integers to define weighted norms and to use the knowledge of the bit length of entries of our target vector, such as [14] (it is possible to know the bit length of entries of our target vector, and this point is the second weakness of DEC).

Our experimental results in Sect. 5 show that one can find correct vectors in the first step with probability being about from 70 to 90 % for the recommended parameters in Sect. 3 via the weighted LLL algorithm. 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 (see [12, Sect. 5]), we infer that our attack can break DEC with sufficiently high probability in polynomial time for all practical parameters.

Notation. For a finite subset Λ of $\mathbb{Z}^n$ and a polynomial $f(x_1, \ldots, x_n) = f(\underline{x}) = \sum_{\underline{i} \in \Lambda} f_{\underline{i}} \underline{x}^{\underline{i}} \in \mathbb{Z}[\underline{x}]$ ($\underline{x}^{\underline{i}} := x_1^{i_1} \cdots x_n^{i_n}$ for $\underline{i} := (i_1, \ldots, i_n)$), we set $\Lambda_f := \{\underline{i} \in \Lambda \mid f_{\underline{i}} \neq 0\}$. We denote by w_f the total degree of f. For $\underline{i} \in \Lambda_f$, we write $c_{\underline{i}}(f)$ as the coefficient of the monomial $\underline{x}^{\underline{i}}$ in f. We set $H(f) := \max_{\underline{i} \in \Lambda_f} |c_{\underline{i}}(f)|$. We define $\mathbf{f} := \left(c_{\underline{i}_1}(f), \ldots, c_{\underline{i}_{\sharp \Lambda_f}}(f)\right)$ if we set Λ_f as $\{\underline{i}_1, \ldots, \underline{i}_{\sharp \Lambda_f}\}$.

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

The formal definitions of weighted norms and weighted lattices are as follows:

Definition 1. For $\mathbf{w} = (w_1, \ldots, w_m) \in (\mathbb{R}_{>0})^m$, we define the norm $\| \cdot \|_{\mathbf{w}}$ on $\mathbb{R}^m$ by $\|\mathbf{a}\|_{\mathbf{w}} := ((a_1 w_1)^2 + \cdots + (a_m w_m)^2)^{1/2} \quad (\mathbf{a} = (a_1, \ldots, a_m) \in \mathbb{R}^m)$. Then 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 $\mathbf{w} \in (\mathbb{R}_{>0})^m$, we denote $\mathcal{L}$ by $\mathcal{L}^{\mathbf{w}}$ if we consider $\mathcal{L}$ as a weighted lattice for $\mathbf{w}$.

Let $\mathcal{L} \subset \mathbb{R}^m$ be a lattice and $\mathbf{w} = (w_i)_{1 \le i \le m} \in (\mathbb{R}_{>0})^m$ a vector. We set W as the diagonal matrix whose (i, i)-entry coincides with w_i for each $1 \le i \le m$. We define the isomorphism $f_W : \mathbb{R}^m \to \mathbb{R}^m$ by $\mathbf{x} \mapsto \mathbf{x}W$. Then for any $\mathbf{x} \in \mathcal{L}^{\mathbf{w}}$, $\mathbf{x}$ is a shortest vector in $\mathcal{L}^{\mathbf{w}}$ if and only if $\mathbf{x}W$ is a shortest vector in $f_W(\mathcal{L})$ with respect to the Euclidean norm. This means 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

We here define a weighted LLL reduced basis and give an algorithm to find it.

Definition 2. Let $\mathcal{L}$, W and f_W be as in Sect. 2.1. We call an ordered basis $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ of the lattice $\mathcal{L}$ a *weighted LLL reduced basis* of $\mathcal{L}$ 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.

Algorithm 1. *Input*: a vector $\mathbf{w} \in (\mathbb{R}_{>0})^m$ and a basis $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ of a lattice $\mathcal{L} \subset \mathbb{R}^m$.
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 $f_W(\mathcal{L})$ (W and f_W are as above).
(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 by using $\{\mathbf{b}_1 W, \ldots, \mathbf{b}_n W\}$.
(3) Compute $\mathbf{b}_i' := \mathbf{b}_i^{\mathbf{w}} W^{-1}$ $(1 \le i \le n)$, and return $\mathcal{B}' = \{\mathbf{b}_1', \ldots, \mathbf{b}_n'\}$.

In our cryptanalysis (Sect. 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.

3 Overview of DEC

In this section, we give a brief review of DEC (see [23, Sect. 3] for details). 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 (see [12, Remark 3.5.1(3)]) with 128 bit security. Note that 1,200 bits is about 10 times smaller than sizes of public keys of efficient candidates of PQC [21,22,25].

3.1 Polynomials of Degree Increasing Type

Polynomials of degree increasing type which play a central role in DEC.

Definition 3. *A polynomial* $X(\underline{x}) \in \mathbb{Z}[\underline{x}] \setminus \{0\}$ *is of degree increasing type if a map* $\Lambda_X \to \mathbb{Z}_{\geq 0}$; $(i_1, \dots, i_n) \mapsto \sum_{k=1}^n i_k$ *is injective.*

Let $X(\underline{x})$ be a polynomial in $\mathbb{Z}[\underline{x}] \setminus \{0\}$. If X is of degree increasing type, then Λ_X is a totally ordered set by the following order $\succ$: for two elements $(i_1, \dots, i_n)$ and $(j_1, \dots, j_n)$ in Λ_X, we have $(i_1, \dots, i_n) \succ (j_1, \dots, j_n)$ 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 above.

Now, we describe only the Key Generation and the Encryption processes of DEC according to [23] since only these two processes are needed to describe our attack (for the decryption, see [23, Sect. 3.4]). Note that although in [23] the security parameter is not suggested, here let us set the security parameter λ.

3.2 Key Generation

Secret Key : A vector $\underline{a} := (a_1, \dots, a_n) \in \mathbb{Z}^n$.
Public Key : (1) An integer d with $\gcd(a_i, d) = 1$ for all i.
(2) An integer e with $\gcd(e, \varphi(d)) = 1$, where φ is the Euler function.
(3) An irreducible polynomial $X \in \mathbb{Z}[\underline{x}]$ of degree increasing type with $X(a_1/d, \dots, a_n/d) = 0$ and $\sharp\Lambda_X \leq w_X$.

For a choice of X and sizes of a_i, e, d and N $(1 \leq i \leq n)$, see [12, Sect. 3.5].

3.3 Encryption

Plaintext : A polynomial $m \in \mathbb{Z}[x_1, \dots, x_n]$ with $\Lambda_m = \Lambda_X$, $1 < c_{i_1,\dots,i_n}(m) < d$ and $\gcd(c_{i_1,\dots,i_n}(m), d) = 1$ for all $(i_1, \dots, i_n) \in \Lambda_m$.

Encryption Process:

(1) Choose a random integer $N \in \mathbb{Z}_{>0}$ uniformly such that $Nd > 2^\lambda H(X)$. For an upper bound of N, see [12, Sect. 3.5].
(2) Construct the twisted plaintext $\tilde{m}(\underline{x}) \in \mathbb{Z}[\underline{x}]$ by putting $\Lambda_{\tilde{m}} := \Lambda_m$ and $c_{\underline{i}}(\tilde{m}) := c_{\underline{i}}(m)^e \pmod{Nd}$ $(0 < c_{\underline{i}}(\tilde{m}) < Nd,\ \underline{i} \in \Lambda_{\tilde{m}})$.
(3) Choose a random $f(\underline{x}) \in \mathbb{Z}[\underline{x}]$ uniformly such that $\Lambda_f = \Lambda_X$ and $H(\tilde{m}) < c_{\underline{k}}(f) < Nd$ 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 $\Lambda_X = \Lambda_{s_j} = \Lambda_{r_j}$ for $1 \leq j \leq 3$ so that $\left|c_{\underline{i}}(s_j)\right|$ and $\left|c_{\underline{i}}(r_j)\right|$ have the same bit length as $\left|c_{\underline{i}}(X)\right|$ and $\left|c_{\underline{i}}(f)\right|$ for each $\underline{i} \in \Lambda_X$, respectively.
(5) Put cipher polynomials $F_1(\underline{x})$, $F_2(\underline{x})$ and $F_3(\underline{x})$ as follows:

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

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

4 Attack Against DEC via Weighted LLL Reduction

In this section, we present an attack against DEC. Main ideas of our attack are the following: (1) Reduce the one-wayness of DEC to finding some appropriate vectors by linearization technique. (2) Find the most important vector for our attack by Algorithm 1 described in Sect. 2.2.

4.1 Algorithm of Our Attack

In this subsection, we write down an algorithm of our attack against DEC (see [12, Sect. 4, Sect. 5] for a more detailed description and complexity analysis of our algorithm below).

Recall from Sects. 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 $F_j = \tilde{m} + s_j f + r_j X$ for $1 \leq j \leq 3$, where $\tilde{m}$, f, s_j and r_j are the twisted plaintext and random polynomials chosen uniformly according to Sect. 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 Sect. 3.1. Let $\underline{k}$ be the maximal element in Λ_X. Note that it is sufficient for recovering the plaintext m to recover $\tilde{m}$, and that $\Lambda_X = \Lambda_m = \Lambda_{\tilde{m}} = \Lambda_f = \Lambda_{s_j} = \Lambda_{r_j}$ for $1 \leq j \leq 3$. This condition allows us to assume $\Lambda_{F_1} = \Lambda_{F_2} = \Lambda_{F_3}$. The algorithm of our attack is as follows:

Algorithm 2. *Input*: a public key (d, e, X) and a ciphertext (F_1, F_2, F_3, N). *Output*: a twisted plaintext $\tilde{m}$.

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

Step 1-1: Put $F'_j := F_j - F_{j+1}$, $r'_j := r_j - r_{j+1}$ and $g := s'_2 r'_1 - s'_1 r'_2$ for $j = 1$ and 2. Solve the linear system obtained by comparing the coefficients of the equality $s'_2 F'_1 - s'_1 F'_2 = gX$. Let $\mathcal{L}'_1$ be the nullspace and $\{\mathbf{u}'_1, \mathbf{u}'_2, \mathbf{u}'_3\}$[1] a lattice basis of $\mathcal{L}'_1$.

Step 1-2: Let $\mathbf{u}_i$ be the vector consisting of the 1-($2\sharp\Lambda_X$)-th entries of $\mathbf{u}'_i$ for $1 \leq i \leq 3$. Let $H(X) := \max_{1 \leq j \leq q} |c_{\underline{i}_j}(X)|$, where $c_{\underline{i}_j}(X)$ is the non-zero coefficient of X for each $\underline{i}_j \in \Lambda_X$ (see Sect. 3.1). Put $w'_j = 2^{\left\lfloor \log_2 \left(\frac{H(X)}{c_{\underline{i}_j}(X)} \right) \right\rfloor}$ for $1 \leq j \leq q$. We then set $\mathbf{w} := (w'_1, \ldots, w'_q, w'_1, \ldots, w'_q)$. Execute Algorithm 1 for the weight $\mathbf{w}$ to the lattice $\mathcal{L}_1 := \langle \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3 \rangle_{\mathbb{Z}}$, 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 $F'_j = s'_j f + r'_j X$ for $j = 1$ and 2, where B is a $(3\sharp\Lambda_X \times \sharp\Lambda_{X^2})$-matrix. Let $\mathbf{v}_0$ be a solution of $\mathbf{v}B = \mathbf{b}$, $\mathcal{L}_2$ the nullspace of $\mathcal{L}_2$ and $\{\mathbf{v}_1\}$ a lattice basis of $\mathcal{L}_2$. Note that if $\gcd(X, s'_1) = 1$, then the rank of $\mathcal{L}_2$ is always equal to 1, see [12, Remark 4.1.4].

[1] We may assume rank $(\mathcal{L}'_1)$ = rank $(\mathcal{L}_3) = 3$ (see [12, Remark 6.0.1]).

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-($\sharp \Lambda_X$)-th entries of $\mathbf{v}_0'$. Construct $f' \in \mathbb{Z}[\underline{x}]$ so that $\mathbf{f}' = \mathbf{v}_0''$, where recall that $\mathbf{f}' := (c_{\underline{i}_1}(f), \ldots, c_{\underline{i}_q}(f))$. Experimentally $\mathbf{v}_0'$ provides the polynomial closer to the correct f than $\mathbf{v}_0$ in many cases.

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

Step 3-1: Solve the linear system $\mathbf{w}C = \mathbf{c}$ obtained by comparing the coefficients of the equality $F_1 = \tilde{m} + s_1 f' + r_1 X$, where C is a $(3\sharp\Lambda_X \times \sharp\Lambda_{X^2})$-matrix. Let $\mathbf{w}_0$ be a solution of $\mathbf{w}C = \mathbf{c}$, $\mathcal{L}_3$ the nullspace of C and $\{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3\}^6$ a lattice basis of $\mathcal{L}_3$, where we take $\mathbf{w}_3$ so that its 1-$\sharp\Lambda_X$-th entries are equal to 0, see [12, Remark 4.1.4].

Step 3-2: Execute Babai's nearest plane algorithm [5] to find a closest vector $\mathbf{z}$ in the lattice $\mathcal{L}_3' := \langle \mathbf{w}_1, \mathbf{w}_2 \rangle_{\mathbb{Z}}$ to $\mathbf{w}_0 + \mathbf{w}_3$ (see [12, Remark 4.1.7] for the reason of using Babai's nearest plane algorithm). Obtain $\mathbf{s}_1$ as the vector consisting of the $(\sharp\Lambda_X + 1)$-$2\sharp\Lambda_X$-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 $F_1 - \tilde{m} - s_1 f' = rX$, where the coefficients of $\tilde{m}$ and r are variables and H is a $(2\sharp\Lambda_X \times \sharp\Lambda_{X^2})$-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 $t \in \mathbb{Z}$. We can compute t by using modular arithmetic and the fact that the coefficient $c_{\underline{k}}$ of X is divisible by d. Finally output $\tilde{m}$.

Remark 1. It may be effective to use Babai's nearest plane algorithm with respect to a weighted norm and to search a desired vector $\mathbf{s}_1$ by adding some vectors in $\mathcal{L}_3'$. However, we will see in Sect. 5, our attack can break the one-wayness of DEC with sufficiently high probability. Thus we omit these processes.

5 Experimental Results on Our Attack

We show some pieces of experimental results[2] on our attack described in Sect. 4 against DEC for $n = 4$, i.e., the number of variables of a public key X is is equal to 4 (see [12, Table 1] for complete experimental results). We conducted experiments for parameters recommended in [12, Remark 3.5.1] which can let the DEC have 128 bit security.

Experimental Procedure

Given parameters w_X and $\sharp\Lambda_X$, we repeat the following procedure 100 times:

1. Make a secret key and a public key (see Sect. 3.2).
2. Using the public key constructed above, make a ciphertext (see Sect. 3.3).
3. Execute Algorithm 2 for the above public key and the ciphertext.

[2] We use a standard note PC with 2.60 GHz CPU (Intel Corei5), 16 GB memory and Mac OS X 64 bit. We implemented the attack in Magma V2.21-3 [8].

In the second step above, we construct a public key X such that $2^{99} \leq |c_{\underline{i}}(X)| < 2^{100}$ 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 Sect. 3.1. In order to show the effect of the weighted LLL algorithm on our cryptanalysis, we also execute another version of Algorithm 2. (The other version of Algorithm 2 here means that the usual LLL algorithm is adopted in Step 1-2 of Algorithm 2 instead of the weighted LLL algorithm.) We count the number of successes and time if $\tilde{m}$ or $-\tilde{m}$ 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 2 in Sect. 4.1). In Step 3 of Table 1, we show the number of successes only if a twisted plaintext $\tilde{m}$ or $-\tilde{m}$ is recovered.

From Step 1 in Table 1, we see that the weighted LLL algorithm found the target vector in Step 1 with probability being about from 70 to 90%, while the usual LLL algorithm could not find the target vector at all. From Step 3 in Table 1, we see that our attack with the weighted LLL algorithm could recover $\tilde{m}$ or $-\tilde{m}$ with probability being about from 20 to 40%, while another attack with the usual LLL algorithm could not recover $\tilde{m}$ or $-\tilde{m}$ 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.

Table 1. Experimental results on Algorithm 2 in Sect. 4.1 against DEC with four variables of 128 bit security. We did experiments according to Experimental Procedure described in the beginning of Sect. 5. "Ave. Time" means the average of time for performing our attack. (We show the timing data in successful cases.)

Recommended parameters for DEC ([12, Remark 3.5.1])			Experimental results					
Total degree of a public key X	Number of monomials of X	The maximum sizes of the coefficients of X except its constant and maximal terms	Number of successes of Attack Algorithm / 100					
			Method for lattice reduction in Step 1					
			Usual LLL			Weighted LLL		
			Step 1	Step 3	Ave. Time	Step 1	Step 3	Ave. Time
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

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 Sect. 6 of [23]). 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.

6 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). It is showed in this paper 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 is not necessarily shortest in a lattice of low rank but only some entries are relatively small. The usual LLL algorithm with respect to well-known norms 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 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 for all practical parameters under some assumptions.

Acknowledgements. The authors thank anonymous referees for careful reading our manuscript and for giving helpful comments. The authors also thank Steven Galbraith, Masaya Yasuda and Shun'ichi Yokoyama for helpful comments. This work was supported by CREST, JST.

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/
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.) PKC 2009. LNCS, vol. 5443, pp. 425–442. Springer, 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, Heidelberg (2009)
8. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. J. Symbol. Comput. **24**(3–4), 235–265 (1997)
9. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. J. Cryptology **10**(4), 233–260 (1997). Springer
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, vol. 25. Springer, US (2006)
12. Ding, J., Kudo, M., Okumura, S., Takagi, T., Tao, C.: Cryptanalysis of a public key cryptosystem based on Diophantine equations via weighted LLL reduction, IACR Cryptology ePrint Archive, 2015/1229 (2015)
13. Davis, M., Matijasevič, Y., Robinson, J.: Hilbert's tenth problem, Diophantine equations: positive aspects of a negative solution. In: Mathematical Developments Arising from Hilbert Problems, pp. 323–378. American Mathematical Society, Providence (1976)
14. Faugère, J.-C., Goyet, C., Renault, G.: Attacking (EC)DSA given only an implicit hint. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 252–274. Springer, Heidelberg (2013)
15. Faugère, J.-C., Spaenlehauer, P.-J.: Algebraic cryptanalysis of the PKC'2009 algebraic surface cryptosystem. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 35–52. Springer, Heidelberg (2010)
16. Galbraith, S.D.: Mathematics of Public Key Cryptography. Cambridge University Press, Cambridge (2012)
17. Hirata-Kohno, N., Pethö, A.: On a key exchange protocol based on Diophantine equations. Infocommun. J. **5**(3), 17–21 (2013). Scientific Association for Infocommunications (HTE)
18. Iwami, M.: A reduction attack on algebraic surface public-key cryptosystems. In: Kapur, D. (ed.) ASCM 2007. LNCS (LNAI), vol. 5081, pp. 323–332. Springer, Heidelberg (2008)
19. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. **261**(4), 515–534 (1982). Springer
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.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, 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. In: Proceedings of the IEEE International Symposium on Information Theory (2013)
23. Okumura, S.: A public key cryptosystem based on diophantine equations of degree increasing type. Pac. J. Math. Ind. **7**(4), 33–45 (2015). Springer, Heidelberg
24. Pheidas, T.: Hilbert's Tenth Problem for fields of rational functions over finite fields. Inventiones Math. **103**(1), 1–8 (1991). Springer

25. Tao, C., Diene, A., Tang, S., Ding, J.: Simple matrix scheme for encryption. In: Gaborit, P. (ed.) PQCrypto 2013. LNCS, vol. 7932, pp. 231–242. Springer, Heidelberg (2013)
26. 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)
27. Videla, C.R.: Hilbert's Tenth Problem for rational function fields in characteristic 2. Proc. Am. Math. Soc. **120**(1), 249–253 (1994). American Mathematical Society
28. Voloch, F.: Breaking the Akiyama-Goto cryptosystem. In: Contemporary Mathematics, Arithmetic, Geometry, Cryptography and Coding Theory, vol. 487, pp. 113–118. American Mathematical Society, Providence (2007)
29. Yosh, H.: The key exchange cryptosystem used with higher order Diophantine equations. Int. J. Netw. Secur. Appl. J. **3**(2), 43–50 (2011)