Okumura *Pacific Journal of Mathematics for Industry* (2015) 7:4
DOI 10.1186/s40736-015-0014-4

Pacific Journal of Mathematics for Industry
a SpringerOpen Journal

ORIGINAL ARTICLE Open Access

A public key cryptosystem based on diophantine equations of degree increasing type

Shinya Okumura

Abstract

In this paper we propose a new public key cryptosystem based on diophantine equations which we call of degree increasing type. We use an analogous method to the "Algebraic Surface Cryptosystem" (ASC) proposed by Akiyama, Goto and Miyake. There are two main differences between our cryptosystem and ASC. One of them is to twist a plaintext by using some modular arithmetic to increase the number of candidates of the plaintext in order to complicate finding the correct plaintext. Another difference is to use a polynomial of degree increasing type to recover the plaintext uniquely even if the plaintext was twisted. Although we have not been able to give a security proof, we give some discussions on how secure our cryptosystem is against known attacks including the ideal decomposition attack, which can break the one-wayness of ASC.

Keywords: Diophantine equation; Post quantum cryptography; Public key cryptography

1 Introduction

After Diffie and Hellman proposed the concept of public key cryptography [11], the theory of cryptography has been developed rapidly and has contributed to the security of networks. This cryptosystem is based on computationally hard problems, for example factorization of large integers and computation of discrete logarithm in large finite groups. The most famous public key cryptosystems are the RSA cryptosystem [27] and elliptic curve cryptosystem [17,22]. Although these cryptosystems have been studied by many researchers, efficient attacks have not been found in general. However, Shor showed that factorization of integers and computation of discrete logarithm are done efficiently by using quantum computers [28]. So it is important to find new computationally hard problems which are intractable even with quantum computers and can be used to construct cryptosystems. We expect that the diophantine problem is one of such problems. This problem is to find integral or rational solutions of a given multivariate polynomial with integer coefficients. Despite many researchers' endeavor (see e.g. [14]), this problem is usually a very difficult problem. Moreover Matijasevič showed that there is no general method which determines the solvability of an arbitrary diophantine equation [10]. On the other hand, for any integers $a_1, a_2, \cdots, a_n$, it is easy to find a polynomial $X(x_1, \cdots, x_n) \in \mathbb{Z}[x_1, \cdots, x_n]$ with $X(a_1, a_2, \cdots, a_n) = 0$ (see section 3.4.1). So we can expect that diophantine equations can be used to construct a new public key cryptosystems. Indeed some cryptosystems based on this problem have already been proposed [15,19,34]. But the one-wayness of the cryptosystem proposed in [19] was broken [9]. On the other hand, cryptosystems in [15,34] are interesting in theory, but these cryptosystems can be used only a few times with the same key ([15], Proposition 2).

We can also consider the diophantine problem over global function fields. This problem is also hard and it is proved that there is no general method which determines the solvability of an arbitrary diophantine equation [25]. The Algebraic Surface Cryptosystem (ASC) proposed in [1] is based on the hardness of the section finding problem (SFP) which can be viewed as a diophantine problem over $\mathbb{F}_p[t]$ (or $\mathbb{F}_p(t)$). More precisely, let p be a prime number and $X(x, y, t) \in \mathbb{F}_p[x, y, t]$ a polynomial which defines

Correspondence: s-okumura@imi.kyushu-u.ac.jp
Kyushu University, 744, Motooka, Nishi-ku, 819-0395 Fukuoka, Japan

a surface S with a fibration $S \to \mathbb{A}^1_{\mathbb{F}_p}$ over the affine t-line. The SFP is to find $u_x(t), u_y(t) \in \mathbb{F}_p[t]$ such that $X(u_x(t), u_y(t), t) = 0$.

In number theory, there are many analogous problems between number fields and function fields. There are many cases where problems over function fields have been solved while the corresponding problems have hardly been solved. For example, there is an algorithm to factorize elements of $\mathbb{F}_p[t]$ in probabilistic polynomial time [2,7], while the best known algorithm (the general number field sieve) for fuctorization in $\mathbb{Z}$ takes subexponential time $O\left(e^{(c+o(1))(\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}}}\right)$, where $c = \left(\frac{9}{64}\right)^{\frac{1}{3}}$ and N is an integer which we want to factorize [18]. The Riemann Hypothesis for function fields was proved by André Weil [33], while the Riemann Hypothesis for $\mathbb{Z}$ still seems far beyond our reach. The *abc* conjecture for function fields (the Mason-Stothers Theorem) was proved in [21,29], while a proof of the *abc* conjecture for $\mathbb{Z}$ was announced just a few years ago by Shinichi Mochizuki [23].

In this paper we consider diophantine equations of degree increasing type (see Definition 3.1) over integers and propose a new public key cryptosystem whose security relies on the hardness to find a rational solution to them. In our cryptosystem we use a polynomial $X(x_1, \cdots, x_n) \in \mathbb{Z}[x_1, \cdots, x_n]$ and integers $d, e \in \mathbb{Z}$ satisfying certain conditions as public keys and integers $a_1, \cdots, a_n$ satisfying $X\left(\frac{a_1}{d}, \cdots, \frac{a_n}{d}\right) = 0$ as secret keys. Our method is to mix a plaintext (this is a polynomial) with other polynomials and cover the mixed polynomial with public key. To recover the plaintext we use secret keys and some modular arithmetic. This method is analogous to ASC except for using modular arithmetic. Although the one-wayness of ASC was broken by the ideal decomposition attack [12], our analysis (section 4) shows that our cryptosystem has resistance against some possible attacks including the ideal decomposition attack. However, we have not been able to give a security proof of it. Finally, we estimate the size of keys of our cryptosystem. This paper aims to design a scheme with 128 bit-security level. Our estimation shows that if we use integers d, e and a diophantine equation with n variables and total degree w as the public key, then the size of the secret key is at most $\left(\lceil\frac{128}{n-1}\rceil + 1\right)n + \lceil\log_2 d - \log_2\varphi(d)\rceil$ bits and the size of the public key is at most $\left(\lceil\frac{128}{n-1}\rceil + 76 + \lceil\log_2 d - \log_2\varphi(d)\rceil\right)w + 65 + \lceil\log_2 e\rceil$ bits. We also estimate the size of ciphertexts to be at most $\frac{3}{2}(w^2+w)(129+130w+\lceil\log_2 w\rceil) + 129 + 65(w-1)$ bits.

This paper is organized as follows: In section 2 we give a brief review of ASC and known attacks against it. In section 3 we describe our cryptosystem including some remarks on it and give a method to construct a diophantine equation of degree increasing type with a given solution. In section 4 we analyze its security against some possible attacks. In section 5 we estimate the size of keys and ciphertexts under some assumptions. In section 6 we give some examples of the size of keys and ciphertexts together with the time which it took to encrypt and decrypt.

2 Review of ASC

In this section we give a brief review of ASC and known attacks against it (for details, see [1]). Let p be a prime number. The ASC makes use of a section to a fibration of an algebraic surface to the afine line over $\mathbb{F}_p$.

2.1 Notation

Let p be a prime number and $\mathbb{F}_p$ a finite field with p elements. For a polynomial $g = \sum_{i,j} g_{ij}(t)x^i y^j = \sum_{i,j,k} g_{ijk}x^i y^j t^k \in \mathbb{F}_p[x,y,t]$ we define

$$\Lambda_g^{(p)} := \{(i,j) \in \mathbb{Z}^2 \mid g_{ij}(t) \neq 0\},$$
$$\Gamma_g^{(p)} := \{(i,j,k) \in \mathbb{Z}^3 \mid g_{ijk} \neq 0\}.$$

For two subsets $\Lambda_1, \Lambda_2 \subset (\mathbb{Z}_{\geq 0})^2$ we define

$$\Lambda_1\Lambda_2 := \{(i_1+i_2, j_1+j_2) \mid (i_1,j_1) \in \Lambda_1, (i_2,j_2) \in \Lambda_2\}.$$

This means that if $\Lambda_i^{(p)} = \Lambda_{f_i}^{(p)}$ for some $f_i \in \mathbb{F}_p[x,y,t]$, then $\Lambda_1\Lambda_2 = \Lambda_{f_1 f_2}^{(p)}$. For each ideal $J = (f_1, \ldots, f_n) \subset \mathbb{F}_p[x,y,t]$, each polynomial $g \in \mathbb{F}_p[x,y,t]$ and each monomial ordering $<$, there are polynomials h, $r \in \mathbb{F}_p[x,y,t]$ such that $h \in J$, $g = h + r$ and that no monomial of r is in the ideal generated by the leading monomials of f_i for $i = 1, \ldots, n$. The r may depend on the choice of a system of generators of J, but is uniquely determined (for a fixed monomial ordering of $\mathbb{F}_p[x,y,t]$) if we calculate it using a Gröbner basis of J. Then this unique r is called the normal form of g with respect to J and $<$, and we denote it by $NF_J(g)$ (see [8]).

2.2 Key generation

1. Secret key
   Choose two polynomials $u_x(t), u_y(t) \in \mathbb{F}_p[t]$ of degree d.
2. Public key
   For $k = 1, 2, 3$, choose finite subsets $\Lambda_k^{(p)} \subset (\mathbb{Z}_{\geq 0})^2$ and $D_k = \left\{d_{ij}^{(k)} \mid (i,j) \in \Lambda_k^{(p)}\right\} \subset \mathbb{Z}_{\geq 0}$ so that the following holds:

   (i) $\Lambda_2^{(p)} \subset \Lambda_1^{(p)}\Lambda_3^{(p)}$.

   (ii) For any polynomial $f_k = \sum_{(i,j)\in\Lambda_k^{(p)}} f_{ij}^{(k)}(t)x^i y^j \in \mathbb{F}_p[x,y,t]$ $(k = 1,2,3)$ with $\Lambda_{f_k}^{(p)} = \Lambda_k^{(p)}$ and $\deg f_{ij}^{(k)}(t) = d_{ij}^{(k)}$, we have

$$\begin{cases} \deg_x f_1 < \deg_x f_2 < \deg_x f_3, \\ \deg_y f_1 < \deg_y f_2 < \deg_y f_3, \\ \deg_t f_1 < \deg_t f_2 < \deg_t f_3, \\ (\deg_x f_2, \deg_y f_2, \deg_t f_2) \in \Gamma_{f_2}^{(p)}, \\ (\deg_x f_3, \deg_y f_3, \deg_t f_3) \in \Gamma_{f_3}^{(p)}. \end{cases} \tag{1}$$

Construct an $X(x,y,t) = \sum_{(i,j)\in\Lambda_1^{(p)}} c_{ij}(t)x^i y^j \in \mathbb{F}_p[x,y,t]$ such that $X(u_x(t), u_y(t), t) = 0$, $\deg c_{ij}(t) = d_{ij}^{(1)}$ and $c_{ij}(t) \neq 0$ for $(i,j) \in \Lambda_1^{(p)}$. In section 2.5 we give a method to construct such a polynomial. For $i = 1,2,3$, make X, $\Lambda_i^{(p)}$ and D_i public.

2.3 Encryption

Assume that the sender wants to send a polynomial $m(x,y,t) = \sum_{(i,j)\in\Lambda_2^{(p)}} m_{ij}(t)x^i y^j \in \mathbb{F}_p[x,y,t]$ with $\deg m_{ij}(t) = d_{ij}^{(2)}$ for $(i,j) \in \Lambda_2^{(p)}$.

1. For $k = 1,2$, choose random polynomials in $\mathbb{F}_p[x,y,t]$:

$$\begin{aligned} s_k &= \sum_{(i,j)\in\Lambda_1^{(p)}} s_{ij}^{(k)}(t)x^i y^j, \\ r_k &= \sum_{(i,j)\in\Lambda_3^{(p)}} r_{ij}^{(k)}(t)x^i y^j, \\ f &= \sum_{(i,j)\in\Lambda_3^{(p)}} f_{ij}(t)x^i y^j, \end{aligned}$$

such that $\deg s_{ij}^{(k)}(t) = d_{ij}^{(1)}$ and $\deg r_{ij}^{(k)}(t) = \deg f_{ij}(t) = d_{ij}^{(3)}$. Note that from (1), we have

$$\begin{cases} \deg_x X < \deg_x m < \deg_x f, \\ \deg_y X < \deg_y m < \deg_y f, \\ \deg_t X < \deg_t m < \deg_t f, \\ (\deg_x m, \deg_y m, \deg_t m) \in \Gamma_m^{(p)}, \\ (\deg_x f, \deg_y f, \deg_t f) \in \Gamma_f^{(p)}. \end{cases} \tag{2}$$

2. Put $F_i := m + s_i f + r_i X$ for $i = 1,2$, and send (F_1, F_2).

2.4 Decryption

1. For $i = 1,2$, compute

$$\begin{aligned} h_i(t) &:= F_i(u_x(t), u_y(t), t) \\ &= m(u_x(t), u_y(t), t) \\ &\quad + s_i(u_x(t), u_y(t), t) f(u_x(t), u_y(t), t). \end{aligned}$$

2. Factorize $h_1 - h_2$ and find a factor h_3 of it whose degree is equal to $\deg f(u_x(t), u_y(t), t)$. Note that from (2), we have

$$\deg f(u_x(t), u_y(t), t) = \deg h_3 > \deg m(u_x(t), u_y(t), t).$$

3. Compute $h_4(t) := h_1(t) \pmod{h_3(t)}$. Note that if h_3 divides $s_1(u_x(t), u_y(t), t) f(u_x(t), u_y(t), t)$, then $h_4 = m(u_x(t), u_y(t), t)$.
4. Extract $m(x,y,t)$ from h_4 by solving the following linear equation

$$h_4 = \sum_{(i,j,k)\in\Gamma_m^{(p)}} m_{ijk} u_x^i u_y^j t^k,$$

in variables m_{ijk} for $(i,j,k) \in \Gamma_m^{(p)}$, and put

$$m'(x,y,t) := \sum_{(i,j,k)\in\Gamma_m^{(p)}} m_{ijk} x^i y^j t^k.$$

5. We can verify whether $m' = m$ or not by a MAC (message authentication code) of m. If the verification fails, then go back to step 2 and choose another factor of $h_1 - h_2$.

2.5 Construction of $X(x,y,t)$

We describe a method to construct a polynomial $X(x,y,t) \in \mathbb{F}_p[x,y,t]$ such that $X(u_x(t), u_y(t), t) = 0$ for given polynomials $u_x(t), u_y(t) \in \mathbb{F}_p[t]$.

1. Choose a finite subset $(0,0) \in \Lambda^{(p)} \subset (\mathbb{Z}_{\geq 0})^2$ and $D := \{(d_{ij} \mid (i,j) \in \Lambda^{(p)}\} \subset \mathbb{Z}_{\geq 0}$.
2. Choose random non-zero polynomials $c_{ij}(t)$ of degree d_{ij} for $(i,j) \in \Lambda^{(p)} \setminus \{(0,0)\}$.
3. Compute $c_{00}(t) := -\sum_{(i,j)\in\Lambda^{(p)}\setminus\{(0,0)\}} c_{ij}(t) u_x^i u_y^j$.
4. Define

$$X := \sum_{(i,j)\in\Lambda^{(p)}} c_{ij}(t)x^i y^j.$$

2.6 Known attacks

We describe four possible attacks against ASC. For more details, see [1], section 5 and [12,24].

2.6.1 Reduction to solving a multivariate equation system

Let

$$\begin{aligned} f'(x,y,t) &= \sum_{(i,j,k)\in\Gamma_f^{(p)}} f'_{ijk} x^i y^j t^k, \\ s'(x,y,t) &= \sum_{(i,j,k)\in\Gamma_{s_1}^{(p)}} s'_{ijk} x^i y^j t^k, \\ r'(x,y,t) &= \sum_{(i,j,k)\in\Gamma_{r_1}^{(p)}} r'_{ijk} x^i y^j t^k, \\ m'(x,y,t) &= \sum_{(i,j,k)\in\Gamma_m^{(p)}} m'_{ijk} x^i y^j t^k, \end{aligned}$$

where f'_{ijk}, s'_{ijk}, r'_{ijk} and m'_{ijk} are variables. If one can get f by solving the following quadratic equation system

$$F_1 - F_2 = (s_1 - s_2)f + (r_1 - r_2)X = s'f' + r'X,$$

then one may get m by solving $NF_I(m') = 0$, where $I = (F_1, f, X) \subset \mathbb{F}_p[x, y, t]$. In [1], it is pointed out that if $\#\Gamma_f^{(p)} > 50$ and $\#\Gamma_{s_1}^{(p)} > 50$, then finding solutions of this system becomes computationally intractable, even if the r'_{ijk}'s are eliminated by substituting rational points of X over a finite extension of $\mathbb{F}_p$.

2.6.2 Reduction attack by Iwami [16]

Since X is made public, one can try to divide $F_1 - F_2$ by X to find f in the remainder. But f does not appear in the remainder because of (2). For this attack, see also [31].

2.6.3 Rational point attack by Voloch [32]

Let $F(x, y, t) := F_1 - F_2$. Let $f'(x, y, t)$ and $s'(x, y, t)$ be as in section 2.6.1. Let $g(x, y, t) := s'(x, y, t)f'(x, y, t)$. We write

$$g(x, y, t) = \sum_{(i,j,k)\in\Gamma_g^{(p)}} g_{ijk}x^i y^j t^k,$$

where g_{ijk} are polynomials in the coefficients of s' and f' for $(i, j, k) \in \Gamma_g^{(p)}$. For a large positive integer L and $\ell = 1, \ldots, L$, if one can find rational points (x_ℓ, y_ℓ, t_ℓ) on $X(x, y, t) = 0$ over a certain extension field of $\mathbb{F}_p$, then one may be able to get $(s_1 - s_2)f$ by solving the following linear equation system

$$g(x_\ell, y_\ell, t_\ell) = F(x_\ell, y_\ell, t_\ell)(\ell = 1, \ldots, L). \tag{3}$$

Then one can find f by factorization and get m as in section 2.6.1. However, one cannot determine f and m uniquely. If $g_0(x, y, t) \in \mathbb{F}_p[x, y, t]$ satisfies (3), then $g_0 + rX$ also satisfies (3) and has the same form as g for any polynomial $r(x, y, t) \in \mathbb{F}_p[x, y, t]$ having the same form as f. In [1], it is pointed out that if $p^{\#\Gamma_r^{(p)}} = p^{\#\Gamma_f^{(p)}} > 2^{100}$, then we may avoid this attack.

2.6.4 Ideal decomposition attack

As mentioned above, we can design ASC to avoid the above three attacks. However, in [12] Faugére and Spaenlehauer proposed a new attack called the ideal decomposition attack and claimed that this attack can fully break ASC. The idea of this attack is to reconstruct the ideal $I := (m, f, X)$ in $\mathbb{F}_p[x, y, t]$ or the ideal $J := (m + z, f, X)$ in $\mathbb{F}_p(t)[x, y, z]$ or $(\mathbb{F}_p[t]/(P(t)))[x, y, z]$ from the data (F_1, F_2, X) by using the ideal decomposition $(F_1 - F_2, X) = ((s_1 - s_2)f, X) = I_1 \bigcap I_2$ for some ideals $I_1 \supset (f, X)$ and I_2. Then the following equality holds:

$$(F_1 + z, F_2 + z, X) + I_1 = (m + z, f, X).$$

(Note that, essentially, a resultant was used to reconstruct J in [12]). Let m' be as in section 2.6.1. Then one can get m by solving $NF_I(m') = 0$ or $NF_J(m' + z) = 0$, where z is a new variable and P is an irreducible polynomial in $\mathbb{F}_p[t]$. There are three versions of this attack called the Level 1, the Level 2 and the Level 3 attack, respectively. The largest difference between these attacks is the polynomial ring under consideration. In the Level 1 attack, the polynomial ring $\mathbb{F}_p[x, y, t]$ is used, and they gave an algorithm to reconstruct the ideal $I \subset \mathbb{F}_p[x, y, t]$. The most time consuming computation in this attack is to compute a Gröbner basis of I to solve $NF_I(m') = 0$. In [12], it is pointed out that the Level 1 attack is not efficient and cannot break ASC for the recommended parameters. In the Level 2 attack, the polynomial ring $\mathbb{F}_p(t)[x, y, z]$ is used. In this case they gave an algorithm (which is similar to the Level 1 attack) to reconstruct the ideal $J \subset \mathbb{F}_p(t)[x, y, z]$. Note that the new variable z is necessary because the ideal (m, f, X) is generically equal to $\mathbb{F}_p(t)[x, y]$ (see [12] section 3.2). The key which accelerates the computation of Gröbner basis is the following observation: the polynomials occuring in ASC have a high degree in t and a low degree in x and y. Thus, it is natural to regard these polynomials as elements of $\mathbb{F}_p(t)[x, y]$ rather than elements of $\mathbb{F}_p[x, y, t]$. To make this attack more practical, in the Level 3 attack a modular arithmetic was used, i.e., the polynomial ring $(\mathbb{F}_p[t]/(P(t)))[x, y, z]$ is used for an irreducible polynomial $P(t)$ with $\deg P > \deg_t m$. The degree in t of the polynomials appearing in the computation of Gröbner basis is bounded by $\deg P(t)$ and so using a polynomial P of small degree, for example $\deg P = \deg_t m + 1$, makes this attack becomes more efficient than the Level 2 attack. Moreover, it is also possible to use a polynomial $P(t) = \prod_i P_i(t)$ such that $P_i(t)$'s are distinct irreducible polynomials and $\sum_i \deg P_i > \deg_t m$. In this case for each i we compute $m \pmod{P_i}$ and get m by the Chinese Remainder Theorem. Since $\deg P_i < \deg P$, we may have more efficient attack by using P having an appropriate number of irreducible factors and degree if $\deg_t m$ is large. Now, we give an algorithm of the Level 3 attack.

1. Choose a constant C and an integer $n \approx \deg_t(m) \cdot \log p/C$. Choose n irreducible polynomials $P_1, \ldots, P_n$ of degree $\approx C/\log p$ such that $\sum_{1\le i\le n} \deg P_i > \deg_t m$. Set $i = 1$.
2. Let $K_i := \mathbb{F}_p[t]/(P_i)$.
3. Let $F_k^{(P_i)} := F_k \pmod{P_i}$ and $X^{(P_i)} := X \pmod{P_i}$. Compute $Q(y) := \mathrm{Res}_x(F_1^{(P_i)} - F_2^{(P_i)}, X^{(P_i)}) \in K_i[y]$, the resultant of $F_1^{(P_i)} - F_2^{(P_i)}$ and $X^{(P_i)}$ with respect to x.
4. Factor $Q(y)$ and let $Q_0(y)$ be an irreducible factor of highest degree.
5. Compute a Gröbner basis of the ideal $J := (F_1^{(P_i)} + z, F_2^{(P_i)} + z, X^{(P_i)}, Q_0) \subset K_i[x, y, z]$ with respect to the graded reverse lexicographical ordering.
6. Using the above Gröbner basis, solve the following linear equation system over K_i to get $m^{(P_i)} := m \pmod{P_i}$

$$NF_J(m' + z) = 0,$$

where m' is as above. If the system has no solution, then go back to step 4 and choose another factor of Q.

7. If $i < n$, then replace i by $i + 1$ and go back to step 2.
8. Recover m from $m^{(P_i)}$ by using the Chinese Remainder Theorem.

3 Our cryptosystem

3.1 Notation

We denote by $\mathbb{Z}[\underline{x}] := \mathbb{Z}[x_1, \ldots, x_n]$ the polynomial ring with n variables. For a vector $\underline{i} := (i_1, \ldots, i_n) \in (\mathbb{Z}_{\geq 0})^n$ we write $\underline{x}^{\underline{i}} := x_1^{i_1} \cdots x_n^{i_n}$ and $\sum \underline{i} := \sum_{1 \leq j \leq n} i_j$. For a finite subset $\Lambda \subset (\mathbb{Z}_{\geq 0})^n$ and a polynomial $f = \sum_{(i_1,\ldots,i_n)\in\Lambda} f_{i_1 \cdots i_n} x_1^{i_1} \cdots x_n^{i_n} = \sum_{\underline{i}\in\Lambda} f_{\underline{i}} \underline{x}^{\underline{i}} \in \mathbb{Z}[\underline{x}]$ we define

$$\Lambda_f := \{\underline{i} \in (\mathbb{Z}_{\geq 0})^n \mid f_{\underline{i}} \neq 0\},$$

$$\Gamma_f := \{(\underline{i}, b_{\underline{i}}) \in \Lambda_f \times \mathbb{Z}_{>0} \mid 2^{b_{\underline{i}}-1} \leq |f_{\underline{i}}| < 2^{b_{\underline{i}}}\}.$$

We call Λ_f the support of f. For example, for $f_1 = 5x_1^4x_2^2x_3 - 13x_1^2x_2 + 7x_3 + 2$ and $f_2 = 8x_1^2x_2^2x_3 - 9x_1x_2^2 + 6x_3 - 11$, we have

$$\Lambda_{f_1} := \{(4,2,1), (2,1,0), (0,0,1), (0,0,0)\},$$

$$\Gamma_{f_1} := \{(4,2,1,3), (2,1,0,4), (0,0,1,3), (0,0,0,2)\},$$

$$\Lambda_{f_2} := \{(2,2,1), (1,2,0), (0,0,1), (0,0,0)\},$$

$$\Gamma_{f_2} := \{(2,2,1,4), (1,2,0,4), (0,0,1,3), (0,0,0,4)\}.$$

We denote by w_f the total degree of f. Define

$$H(f) := \max\{|f_{\underline{i}}| \mid \underline{i} \in \Lambda_f\}.$$

For a vector $\underline{v} := (v_1, \ldots, v_n) \in \mathbb{Q}^n$, we denote by $f(\underline{v})$ the value of f at $\underline{v}$. For an integer d, we denote by $\underline{v}/d$ the vector $\left(\frac{v_1}{d}, \ldots, \frac{v_n}{d}\right)$. For each ideal $J \subset \mathbb{Q}[\underline{x}]$, each polynomial $f \in \mathbb{Q}[\underline{x}]$ and each monomial ordering $<$, we denote by $NF_J(f)$ a normal form of f with respect to J and $<$. For a polynomial $f \in \mathbb{Z}[\underline{x}]$ and an integer m, we denote by $f^{(m)}$ the polynomial $f \pmod m \in (\mathbb{Z}/m\mathbb{Z})[\underline{x}]$.

3.2 Polynomials of degree increasing type

Before we describe our cryptosystem, we define the following notion which is one of our key ideas to construct our cryptosystem.

Definition 3.1. Define a map $\sigma : \mathbb{Z}^n \longrightarrow \mathbb{Z}$ by $\underline{i} \mapsto \sum \underline{i}$. A polynomial $X \in \mathbb{Z}[\underline{x}]$ is *of degree increasing type* if $\sigma|_{\Lambda_X}$ is injective. In other words, X is of *degree increasing type* if and only if for each $k \in \mathbb{Z}$, X has at most one term of degree k.

Remark 3.2. We can prove that there is no general algorithm to solve an arbitrary diophantine equation of degree increasing type in $\mathbb{Z}$. This can be seen as follows: Suppose $T \in \mathbb{Z}[\underline{x}]$ is an arbitrary polynomial. It is easy to see that by making a change of variables $x_i \mapsto x_i^{q_i}$ with suitable q_i's, we can make $T\left(x_1^{q_1}, \ldots, x_n^{q_n}\right)$ of degree increasing type. Thus if there exists an algorithm to solve an arbitrary diophantine equation of degree increasing type, then it can solve an arbitrary diophantine equation, which contradicts Matijasevič's result [10].

Example 3.3. *If* $X(x, y) := 5x^3y^2 + 12xy^2 + 7xy + 6x + 5$, *then* X *is of degree increasing type.*

Let $X \in \mathbb{Z}[\underline{x}]$ be a polynomial of degree increasing type. Then we can define a total order in Λ_f as follows: for $\underline{i}_1$, $\underline{i}_2 \in \Lambda_f$, we define $\underline{i}_1 \geq \underline{i}_2$ if $\sum \underline{i}_1 \geq \sum \underline{i}_2$. Since Λ_f is finite, there is a maximal element $\underline{k}$. We call the coefficient of degree $\sum \underline{k}$ of X the *leading coefficient* of X and denote it by $ld(X)$.

3.3 Outline of our cryptosystem

We use an analogous method to ASC. More precisely, we use a polynomial $X(\underline{x}) \in \mathbb{Z}[\underline{x}]$ of degree increasing type and a solution $\underline{a} = \left(\frac{a_1}{d}, \ldots, \frac{a_n}{d}\right) \in \mathbb{Q}^n$ to $X = 0$ as a public key and a secret key, respectively. A plaintext is given as a polynomial $m \in \mathbb{Z}[\underline{x}]$. We use the following polynomials in $\mathbb{Z}[\underline{x}]$ as cipher polynomials in our cryptosystem:

$$F_i(\underline{x}) := \tilde{m} + s_i f + r_i X (i = 1, 2, 3),$$

where $\tilde{m}$, s_i, f and r_i are polynomials in $\mathbb{Z}[\underline{x}]$ with $\Lambda_X = \Lambda_{\tilde{m}} = \Lambda_f = \Lambda_{s_i} = \Lambda_{r_i}$. The polynomial $\tilde{m}$ is constructed from a plaintext polynomial $m \in \mathbb{Z}[\underline{x}]$ and has large coefficients (see section 3.4.2). We need to have $\Lambda_X = \Lambda_{\tilde{m}} = \Lambda_f = \Lambda_{s_i} = \Lambda_{r_i}$ and translate m into $\tilde{m}$ to avoid the ideal decomposition attack and other attacks (see section 4). It is the lagest difference between ASC and our cryptosystem. Recall that w_X is the total degree of X. We compute $h_i := F_i(\underline{a})$, $H_1 := (F_1(\underline{a}) - F_2(\underline{a}))d^{2w_X}$, $H_2 := (F_1(\underline{a}) - F_3(\underline{a}))d^{2w_X}$ and $g := \gcd(H_1, H_2)$ to get $\tilde{m}(\underline{a})d^{w_X}$. We pointed out that unlike factorizing a polynomial in $\mathbb{F}_p[t]$, it is hard to factorize integers and so we use three polynomials as cipher polynomials and a GCD computation to get $f(\underline{a})d^{w_X}$. If $g = |f(\underline{a})d^{w_X}|$ and $g > |\tilde{m}(\underline{a})d^{w_X}|$, then we can get $\tilde{m}(\underline{a})d^{w_X}$ by computing $H := h_1 d^{2w_X} \pmod g$ and $Hd^{-w_X} \pmod g$. If we can get $\tilde{m}(\underline{a})d^{w_X}$, then we can recover m by the Recovering Algorithm (RA) described in section 3.4.4. In order to use RA, $\tilde{m}$ must be of degree increasing type (see section 3.4.4) and for security reasons (section 4), an X must have the same support as $\tilde{m}$. So we use X which is of degree increasing type.

3.4 Algorithm of our cryptosystem

Now, we describe our cryptosystem.

3.4.1 Key generation

1. Secret key
   Choose a vector $\underline{a} = (a_1, \ldots, a_n) \in \mathbb{Z}^n$ of a suitable size[a] such that $\gcd(a_i, d) = 1$ for $i = 1, \ldots, n$. Make them secret.

2. Public key
   Choose integers d and e of suitable sizes[b] such that $\gcd(e, \varphi(d)) = 1$. Choose an irreducible polynomial $X(\underline{x}) \in \mathbb{Z}[\underline{x}]$ of degree increasing type such that $X(\underline{a}/d) = 0$ and $\#\Lambda_X \leq w = w_X$. Make e, X and Λ_X public.

We give a method to construct a public key X of degree increasing type with $X(\underline{a}/d) = 0$.

1. Choose a finite subset $\Lambda \subset (\mathbb{Z}_{\geq 0})^n$ such that $\#\left\{\sum \underline{i} \mid \underline{i} \in \Lambda\right\} = \#\Lambda$.
2. Let $\underline{k} = (k_1, \ldots, k_n)$ be the maximal element of Λ. For $\underline{i} \in \Lambda' := \Lambda \setminus \{\underline{0}, \underline{k}\}$, choose random non-zero integers $c_{\underline{i}}$.
3. Choose $c_{\underline{0}}$ and $c_{\underline{k}}$ so that
$$\frac{c_{\underline{k}}\underline{a}^{\underline{k}} + c_{\underline{0}}d^w}{d^w} = -\frac{\sum_{\underline{i}\in\Lambda'} c_{\underline{i}}\underline{a}^{\underline{i}}d^{w'-\sum\underline{i}}}{d^{w'}},$$
where $w' = \max\left\{\sum \underline{i} \mid \underline{i} \in \Lambda'\right\}$, by solving the linear diophantine equation
$$c_{\underline{k}}\underline{a}^{\underline{k}} + c_{\underline{0}}d^w = -\sum_{\underline{i}\in\Lambda'} c_{\underline{i}}\underline{a}^{\underline{i}}d^{w-\sum\underline{i}}. \tag{4}$$
4. Define
$$X := \sum_{\underline{i}\in\Lambda} c_{\underline{i}}\underline{x}^{\underline{i}}.$$

The condition on Λ (step 1 above) means that X is of degree increasing type. The equation (4) means that $X(\underline{a}/d) = 0$.

3.4.2 Encryption

Assume that the sender wants to send a polynomial $m(\underline{x}) = \sum_{\underline{i}\in\Lambda_m} m_{\underline{i}}\underline{x}^{\underline{i}} \in \mathbb{Z}[\underline{x}]$ ($1 < m_{\underline{i}} < d$ and $\gcd(m_{\underline{i}}, d) = 1$) with $\Lambda_m = \Lambda_X$.

1. Choose a positive integer N such that Nd is larger than the absolute value of each coefficient of X. We assume that an upper bound of N is given.
2. Construct a polynomial $\tilde{m}(\underline{x})$ with $\Lambda_{\tilde{m}} = \Lambda_m$ as follows:
   Let $\tilde{m}_{\underline{i}}$ be an integer such that $0 < \tilde{m}_{\underline{i}} < Nd$ and $\tilde{m}_{\underline{i}} \equiv m_{\underline{i}}^e \pmod{Nd}$, and put $\tilde{m}(\underline{x}) = \sum_{\underline{i}\in\Lambda_m} \tilde{m}_{\underline{i}}\underline{x}^{\underline{i}}$.
3. Choose a random polynomial $f \in \mathbb{Z}[\underline{x}]$ with $\Lambda_f = \Lambda_X$ such that $H(\tilde{m}) < ld(f) < Nd$ and $ld(f)$ is relatively prime to d. We also assume that all coefficients of f except $ld(f)$ are also as large as the coefficients of $\tilde{m}$.
4. Choose random polynomials s_i and r_i in $\mathbb{Z}[\underline{x}]$ with $\Gamma_{s_i} = \Gamma_X$ and $\Gamma_{r_i} = \Gamma_f$ for $1 \leq i \leq 3$.
5. Put $F_i := \tilde{m} + s_i f + r_i X$ for $1 \leq i \leq 3$ and send (F_1, F_2, F_3, N).

3.4.3 Decryption

1. Compute $h_i := F_i(\underline{a}/d) = \tilde{m}(\underline{a}/d) + s_i(\underline{a}/d)f(\underline{a}/d)$, $H_1 := (h_1 - h_2)d^{2w_X}$ and $H_2 := (h_1 - h_3)d^{2w_X}$. Note that $H_1, H_2 \in \mathbb{Z}$.
2. Compute $g := \gcd(H_1, H_2) > 0$, the greatest common divisor of H_1 and H_2. If $\gcd(g, d) > 1$, then we replace g by $\frac{g}{\gcd(g,d)}$. Note that if $g = f(\underline{a}/d)d^{w_X}$, then $\gcd(g, d) = 1$ (cf. Remark 3.6.3).
3. Compute $H := h_1 d^{2w_X} \pmod{g}$ and $\tilde{\mu} := Hd^{-w_X} \pmod{g}$. Note that if $|g| > |\tilde{m}(\underline{a}/d)d^{w_X}|$ and g divides $s_1(\underline{a}/d)f(\underline{a}/d)d^{2w_X}$, then we have
$$\tilde{m}(\underline{a}/d)d^{w_X} = \begin{cases} \tilde{\mu} & \text{if } \tilde{m}(\underline{a}/d)d^{w_X} > 0, \\ \tilde{\mu} - g & \text{if } \tilde{m}(\underline{a}/d)d^{w_X} < 0. \end{cases}$$
   Note that $\tilde{m}(\underline{a}/d)d^{w_X} \neq 0$ (cf. Remark 3.6.4).
4. Recover $m(\underline{x})$ from $\tilde{\mu}$ or $\tilde{\mu} - g$ by RA which we will describe below.

3.4.4 Recovering Algorithm (RA)

We describe a method to recover $m(\underline{x})$ from $\tilde{\mu}$. Let N, d, e and Λ_X be as above.
Input : $\tilde{\mu}$, N, d, e and Λ_X.
Output : $m'(\underline{x}) \in \mathbb{Z}[\underline{x}]$ or "false".

1. Compute
$$e' := e^{-1} (\mathrm{mod}\ \varphi(d)).$$
2. Let $\underline{k}$ be the maximal element of Λ_X. Compute
$$m'_{\underline{k}} := \left(\tilde{\mu}\underline{a}^{-\underline{k}}\right)^{e'} \pmod{d}\ (0 < m'_{\underline{k}} < d),$$
$$\tilde{m}'_{\underline{k}} := \left(m'_{\underline{k}}\right)^{e} \pmod{Nd}\ (0 < \tilde{m}'_{\underline{k}} < Nd).$$
3. If $\Lambda'_X := \Lambda_X \setminus \underline{k} = \emptyset$, then return $m'(\underline{x}) = \sum_{\underline{i}\in\Lambda_X} m'_{\underline{i}}\underline{x}^{\underline{i}}$. Otherwise, let $\underline{k}'$ be the maximal element of Λ'_X. Let $w'_X := \sum \underline{k}'$. Put $\tilde{\mu}' := \frac{\tilde{\mu} - \tilde{m}'_{\underline{k}}\underline{a}^{\underline{k}}}{d^{w_X - w'_X}}$. If $\tilde{\mu}' \in \mathbb{Z}$, then replace $\tilde{\mu}$, $\underline{k}$ and Λ_X by $\tilde{\mu}'$, $\underline{k}'$ and Λ'_X, respectively. Otherwise, return "false".
4. Go back to step 2.

Proposition 3.4. *If* $\tilde{\mu} = \tilde{m}(\underline{a}/d)d^{w_{\tilde{m}}}$, *then RA returns* $m(\underline{x})$.

Proof. We assume that $\tilde{\mu} = \tilde{m}(\underline{a}/d)d^{w_{\tilde{m}}} = ld(\tilde{m})\underline{a}^{\underline{k}} + \sum_{\underline{i}\in\Lambda_X\setminus\{\underline{k}\}} \tilde{m}_{\underline{i}}\underline{a}^{\underline{i}}d^{\sum\underline{k}-\sum\underline{i}}$. Because $\tilde{m}$ is of degree increasing type, we have $\sum \underline{k} - \sum \underline{i} \geq 1$. It implies that
$$m'_{\underline{k}} \equiv ld(\tilde{m})^{e'} \equiv m_{\underline{k}}^{ee'} \equiv m_{\underline{k}} (\mathrm{mod}\ d),$$
$$\tilde{m}'_{\underline{k}} \equiv \tilde{m}_{\underline{k}} (\mathrm{mod}\ Nd).$$

Because $ld(m) < d$, we have

$$m_{\underline{k}} = m'_{\underline{k}},$$
$$\tilde{m}_{\underline{k}} = \tilde{m}'_{\underline{k}}.$$

Thus, $\tilde{\mu}' = \tilde{m}_{\underline{k}'}\underline{a}^{\underline{k}'} + \sum_{\underline{i}\in\Lambda'_X\setminus\{\underline{k}'\}} \tilde{m}_{\underline{i}}\underline{a}^{\underline{i}}d^{\sum\underline{k}'-\sum\underline{i}}$. Because $\tilde{m}$ is of degree increasing type, we have $\sum\underline{k}' - \sum\underline{i} \geq 1$. It implies that we can get $m_{\underline{k}'}$ as above. Similarly, we can get $m_{\underline{i}}$ for $\underline{i} \in \Lambda_X \setminus \{\underline{k}, \underline{k}'\}$. □

Remark 3.5. We give some remarks on our cryptosystem.

1. If $d = p$ is a prime number, we may choose $e = p$ and $e' = 1$.
2. We should choose d so that the computation of $\varphi(d)$ is easy. For example, if d is a prime number, then $\varphi(d) = d - 1$.

3.5 Improvement in recovering algorithm

In step 2 of the decryption process we can write $g = f(\underline{a}/d)d^{w_X}t$ ($t \in \mathbb{Z}$). If $|t| > 1$, then, in step 3, g may not divide $s_1(\underline{a}/d)f(\underline{a}/d)d^{2w_X}$. If so, both $\tilde{\mu}$ and $\tilde{\mu} - g$ are not equal to $\tilde{m}(\underline{a}/d)d^{w_X}$. Then RA will return "false" with high probability because d is large, $\sharp\Lambda_X \leq w_X$ and hence $w_X - w'_X$ becomes ≥ 2 in the middle of the process of RA. In this case we must take the following steps:

1. If RA returned "false", then we choose a positive integer M and construct the set $F(g, M) := \{x \in \mathbb{Z} \mid 2 \leq x \leq M, x|g\} \subset \mathbb{Z}$.
2. If $F(g, M) \neq \emptyset$, then we choose an element $x \in F(g, M)$ and remove x from $F(g, M)$. Otherwise, go back to step 1 and choose an integer which is larger than M.
3. Compute $g' := \frac{g}{x}$, $H' := h_1 d^{2w_X} \pmod{g'}$ and $\tilde{\mu}' := H'd^{-w_X} \pmod{g'}$ and recover $m(\underline{x})$ from $\tilde{\mu}'$.
4. If RA returned "false" again, then go back to step 2.

We describe the reason why RA returns "false" with high probability if we do not get $\tilde{m}(\underline{a}/d)d^{w_X}$. Because $\sharp\Lambda_X = w_X + 1$ implies $w_X - w'_X = 1$, we have always $d^{w_X - w'_X} \mid \left(\tilde{\mu} - \tilde{m}'_{\underline{k}}\underline{a}^{\underline{k}}\right)$. Thus in this case RA does not return "false" even if we do not get $\tilde{m}(\underline{a}/d)d^{w_X}$. On the other hand if $\sharp\Lambda_X \leq w_X$, then $w_X - w'_X \geq 2$ is satisfied in the middle of the process of RA and then RA returns "false" with high probability, if we do not get $\tilde{m}(\underline{a}/d)d^{w_X}$. Thus we need to improve the success probability of decryption.

Remark 3.6.

1. In step 3 of the decryption process, we require that $|g| > |\tilde{m}(\underline{a}/d)d^{w_X}|$ to get $\tilde{m}(\underline{a}/d)d^{w_X}$. To satisfy this condition we impose the condition of step 3 in the encryption process on $ld(f)$. Note that the fact that X is of degree increasing type also helps to satisfy $|g| > |\tilde{m}(\underline{a}/d)d^{w_X}|$, because $O(f) = O(\underline{x}^{\underline{k}}) = O(\tilde{m})$ as $x_1, \ldots, x_n \to \infty$ $\left(\sum\underline{k} = w_X\right)$, if X is of degree increasing type. Thus, if $f_{\underline{k}} > \tilde{m}_{\underline{k}}$ and $|a_1|, \ldots, |a_n| \gg d$, then $|f(\underline{a}/d)d^{w_X}| > |\tilde{m}(\underline{a}/d)d^{w_X}|$ is satisfied with high probability because $|\frac{a_1}{d}|, \ldots, |\frac{a_n}{d}| \gg 1$. We also note that we can estimate whether $\tilde{m}(\underline{a}/d)d^{w_X} > 0$ or not by the same reason with high probability.
2. If $|a_1|, \ldots, |a_n| \approx d$ or $|a_1|, \ldots, |a_n| \ll d$, then the argument in Remark 3.6.1 is not correct because $|\frac{a_1}{d}|, \ldots, |\frac{a_n}{d}| \approx 1$ or $|\frac{a_1}{d}|, \ldots, |\frac{a_n}{d}| \ll 1$. So in this case $\underline{a}$ and f should be chosen so that $a_1, \ldots, a_n > 0$ and, for each $\underline{i} \in \Lambda_f$, the absolute value of the $\underline{i}$-th coefficient of f is larger than that of the monomial $\underline{x}^{\underline{i}}$ of $\tilde{m}$ to satisfy $|f(\underline{a}/d)d^{w_X}| > |\tilde{m}(\underline{a}/d)d^{w_X}|$.
3. We need to have $\gcd(f(\underline{a}/d)d^{w_X}, d) = 1$ to compute the inverse element of $d \pmod{g}$. We show that this condition is satisfied. Let $\underline{k}$ be the maximal element of Λ_f. It follows from the expression

$$f(\underline{a}/d)d^{w_X} = f_{\underline{k}}\underline{a}^{\underline{k}} + \sum_{\underline{i}\in\Lambda_f\setminus\{\underline{k}\}} f_{\underline{i}}\underline{a}^{\underline{i}}d^{w_X-\sum\underline{i}},$$

that if $\gcd(f(\underline{a}/d)d^{w_X}, d) = d' > 1$, then $f_{\underline{k}}$ is divisible by d' because $\gcd(\underline{a}^{\underline{k}}, d) = 1$ is satisfied, and $\sum_{\underline{i}\in\Lambda_f\setminus\{\underline{k}\}} f_{\underline{i}}\underline{a}^{\underline{i}}d^{w_X-\sum\underline{i}}$ is divisible by d. This contradicts our assumption because we assume $\gcd(f_{\underline{k}}, d) = 1$ in step 3 of the encryption process.
4. We also need to have $\tilde{m}(\underline{a}/d)d^{w_X} \neq 0$ to recover m. We show that this condition is satisfied. Let $\underline{k}$ be as above. It follows from the expression

$$\tilde{m}(\underline{a}/d)d^{w_X} = \tilde{m}_{\underline{k}}\underline{a}^{\underline{k}} + \sum_{\underline{i}\in\Lambda_{\tilde{m}}\setminus\{\underline{k}\}} \tilde{m}_{\underline{i}}\underline{a}^{\underline{i}}d^{w_X-\sum\underline{i}},$$

that if $\tilde{m}(\underline{a}/d)d^{w_X} = 0$, then $\tilde{m}_{\underline{k}}$ is divisible by d. This is a contradiction because $\gcd(m_{\underline{k}}, d) = 1$ implies $\gcd(\tilde{m}_{\underline{k}}, d) = 1$.
5. Recall that the t in section 3.5 is troublesome if it is large. We experimented 100000 times on the value of t for each set of parameters in the following tables. According to these results, we can expect that t is smaller than 1000 with high probability. So we can get $\tilde{m}(\underline{a}/d)d^{w_X}$ in practical time with high probability. However, we do have $t >> 1000$, though it happens with low probability. In this case we would not be able to decrypt the plaintext in practical time by the simple trial. Thus if we want to design a scheme with lower probability of decryption failure, we need an efficient integer factorization algorithm in the above steps Tables 1, 2 and 3.

4 Security analysis

In this section although we have not been able to give a security proof, we analyze the effectiveness of some

Table 1 Quantities of t for $|t| < 100$

No.	n	w_X	$\#\Lambda_X$	$\lvert a_i\rvert$ (bit)	$\lvert t\rvert < 100$ (time)
1	3	5	5	66	99341
2	3	7	7	66	99357
3	3	10	10	66	99398

possible attacks for the one-wayness of our cryptosystem. We also discuss the sizes of d, e and N to achieve 128 bit-security. First, we note that the attacks against ASC described in section 2.6 are applicable also to our cryptosystem.

4.1 Reduction to solving a multivariate equation system I

Let

$$f'(\underline{x}) = \sum_{\underline{i}\in\Lambda_f} f'_{\underline{i}}\underline{x}^{\underline{i}},$$
$$s'(\underline{x}) = \sum_{\underline{i}\in\Lambda_{s_1}} s'_{\underline{i}}\underline{x}^{\underline{i}},$$
$$r'(\underline{x}) = \sum_{\underline{i}\in\Lambda_{r_1}} r'_{\underline{i}}\underline{x}^{\underline{i}},$$

where $f'_{\underline{i}}$, $s'_{\underline{i}}$ and $r'_{\underline{i}}$ are variables. One may be able to get f by solving the following quadratic equation system

$$F_1 - F_2 = (s_1 - s_2)f + (r_1 - r_2)X = s'f' + r'X. \quad (5)$$

The number of variables of the system is smaller than that of the system in section 2.6.1, but experimentally a Gröbner basis of the ideal generated by the coefficients of $F_1 - F_2 - (s'f' + r'X)$ consists of quadratic polynomials and there is no known general algorithm to solve a multivariate quadratic equation system over $\mathbb{Z}$ or $\mathbb{Q}$ in polynomial time. So solving the system would not be easy. Moreover, if $\Lambda_{s_1} = \Lambda_f = \Lambda_{r_1} = \Lambda_X$, then the equalities

$$\begin{aligned} s'f' + r'X &= s'\left(f' + tX\right) + \left(r' - ts'\right)X \\ &= \left(s' + sX\right)f' + \left(r' - sf'\right)X, \end{aligned}$$

where s and t are any integers, show that there are many solutions of the system (5). So we may avoid this attack.

Table 2 Quantities of the t for $|t| < 1000$

No.	n	w_X	$\#\Lambda_X$	$\lvert a_i\rvert$ (bit)	$\lvert t\rvert < 1000$ (time)
1	3	5	5	66	99910
2	3	7	7	66	99929
3	3	10	10	66	99931

Table 3 Quantities of the t for $|t| > 10000$

No.	n	w_X	$\#\Lambda_X$	$\lvert a_i\rvert$ (bit)	$\lvert t\rvert > 10000$ (time)
1	3	5	5	66	32
2	3	7	7	66	12
3	3	10	10	66	11

4.2 Reduction to solving a multivariate equation system II

Let $f'(\underline{x}) = \sum_{\underline{i}\in\Lambda_f} f'_{\underline{i}}\underline{x}^{\underline{i}}$, $s'(\underline{x}) = \sum_{\underline{i}\in\Lambda_{s_1}} s'_{\underline{i}}\underline{x}^{\underline{i}}$ and $r'(\underline{x}) = \sum_{\underline{i}\in\Lambda_{r_1}} r'_{\underline{i}}\underline{x}^{\underline{i}}$ be as in section 4.1. Let

$$\tilde{m}'(\underline{x}) := \sum_{\underline{i}\in\Lambda_{\tilde{m}}} \tilde{m}'_{\underline{i}}\underline{x}^{\underline{i}},$$
$$F' := \tilde{m}' + s'f' + r'X,$$

where $\tilde{m}'_{\underline{i}}$ are variables. Let $\underline{a}_1 = (a_{11}, \ldots, a_{1n}), \cdots, \underline{a}_\ell = (a_{\ell 1}, \ldots, a_{\ell n}) \in \mathbb{Z}^n$ be n-tuples of integers. Then we have the following multivariate equation system in $f'_{\underline{i}}$, $s'_{\underline{i}}$, $r'_{\underline{i}}$ and $\tilde{m}'_{\underline{i}}$:

$$\begin{cases} G_1\left(\tilde{m}'_{\underline{0}}, \cdots, r'_{\underline{k}}\right) := F'(\underline{a}_1) - F_1(\underline{a}_1) = 0 \\ \vdots \\ G_\ell\left(\tilde{m}'_{\underline{0}}, \cdots, r'_{\underline{k}}\right) := F'(\underline{a}_\ell) - F_1(\underline{a}_\ell) = 0. \end{cases} \quad (6)$$

One of the methods of solving (6) is to use the Gröbner basis technique. However, if $\{g_1, \cdots, g_h\}$ is a Gröbner basis of the ideal $(G_1, \cdots, G_\ell)$, experimentally, g_i is a cubic or a quadratic polynomial with rational coefficients having large denominators and numerators. Thus, as mentioned in section 4.1, it would not be easy to solve (6). Moreover, for any integers s and t we have

$$F' = \tilde{m}' + s'(f' + tX) + (r' - ts')X.$$

Noting that $\Lambda_X = \Lambda_{\tilde{m}} = \Lambda_f = \Lambda_{s_1}$, $\Gamma_{s_i} = \Gamma_X$ and $\Gamma_{r_i} = \Gamma_f$ for $1 \le i \le 3$, we see that there are many possible solutions of (6). Hence, we may suppose that this attack is not efficient if Nd is sufficiently large, say $Nd > 2^{128}H(X)$. Note that it is also possible to compare $F'(\underline{a}_i) - \tilde{m}'(\underline{a}_i)$ and $F_1(\underline{a}_i) - F_2(\underline{a}_i)$ to get f, but it would be hard because of the same reason.

4.3 Reduction to solving a multivariate equation system III

The following attack was suggested by Professor Attila Pethő. Let f', s', r' and $\tilde{m}'$ be as in section 4.2. Let $S := \sum_{\underline{i}\in\Lambda_{f's'}} S_{\underline{i}}\underline{x}^{\underline{i}}$ and define

$$F'' := \tilde{m}' + S + r'X,$$

where $S_{\underline{i}}$'s are variables. Then one can apply the similar attack as in section 4.2 to F''. However, we may also suppose that this attack is not efficient if Nd is sufficiently

large, say $Nd > 2^{128}H(X)$. To see this, let $r \in \mathbb{Z}[\underline{x}]$ be a random polynomial with $\Lambda_r = \Lambda_X$. Then we have

$$\begin{aligned} F'' &= \tilde{m}' + (S + rX) + (r' - r)X \\ &= (\tilde{m}' - r) + (S + r) + r'X. \end{aligned}$$

It implies that there are many possible solution to $F''(\underline{a}_1) - F_1(\underline{a}_1) = 0, \ldots, F''(\underline{a}_\ell) - F_1(\underline{a}_\ell) = 0$, where $\underline{a}_1, \ldots, \underline{a}_\ell$ are as in section 4.2. Note that $S + rX$ has the same form as S, and $r' - r$, $\tilde{m}' - r$ and $S + r$ have the same form as r', $\tilde{m}'$ and S, respectively.

4.4 Reduction by X

Since X is made public, one can try to divide $F_1 - F_2$ by X to find f in the remainder. But f does not appear in the remainder if $\Lambda_f = \Lambda_X$ and the absolute values of coefficients of f are larger than those of X. So this attack would not be effective.

4.5 Rational point attack (solving $X = 0$)

This attack is equivalent to solving the diophantine equation $X(\underline{x}) = 0$. Although it is hard in general as mentioned in introduction, one may wonder if the diophantine equation $X(\underline{x}) = 0$ may be solvable for X of degree increasing type. However, there are no known general algorithms to solve such diophantine equations in polynomial time. For instance, in [20], it was proved that the problem for determining whether there are positive integer solutions for

$$ax_1^2 + bx_2 - c = 0,$$

where a, b and c are positive integers, is NP-complete. So we may assume that solving the diopantine equations of degree increasing type is hard in general.

Next, we discuss more general diophantine problems. If one can find a vector $\underline{a}$ such that $X(\underline{a}/d) = 0$, then one can get m by the same process of decryption. The solution $\underline{a}/d$ is not an integral solution but a rational solution. (Using rational solutions is suggested by Professor Noriko Hirata-Kohno.) However, finding such rational solutions is equivalent to finding integral solutions of $G(\underline{x}) := X(\underline{x}/d)d^{w_X} = 0$. (If we do not know the denominator d, finding rational solutions of $G(\underline{x}) = 0$ is reduced to finding integer solutions of the equation $G\left(\frac{x_1}{z}, \ldots, \frac{x_n}{z}\right) z^{w_X} = 0$ in $n + 1$ variables.) If $n = 2$ and $G(\underline{x}) = 0$ defines a curve of genus 0, 1 or a hyperelliptic curve, then there are explicit algorithms to find all integral solutions [6,26,30]. Otherwise, in special cases there are some algorithms to find all integral points [3,4]. Moreover, it is believed that in many cases, diophantine equations with two variables are solvable. Theoretically, using Baker's method and its improvements, explicit upper bounds of the size of solutions to special equations with two variables are known (see [13] and the references given there). (Note that if solutions of a diophantine equation are sufficiently large, then Baker's method is not practical in general, but we want to use a solution which is as small as possible.) However, no efficient methods are known to find integral solutions of diophantine equations of n variables with $n \geq 3$. So we should use a diophatine equations with at least 3 variables as a public key of our cryptosystem. Note that in case of 3 variables, our experience in arithmetic geometry suggests to use X of degree at least 5, because then the hypersurface in the projective 3-space defined by (the homogenized form of) X is of general type if it is non-singular (cf. [14], Example F.5.1.7 and section F.5.2).

4.6 Solving $X(\underline{x}/d)d^{w_X} \equiv 0 \left(\mathrm{modd}^{w_X+1}\right)$

If we use a single cipher polynomial $F := \tilde{m} + rX$, where r is an integer or a polynomial in $\mathbb{Z}[\underline{x}]$ such that rX is of degree increasing type, and $\Lambda_{\tilde{m}} = \Lambda_{rX}$, then it can be broken by finding a solution to the congruence equation

$$X(\underline{x}/d)d^{w_X} \equiv 0 (\mathrm{mod}\ d^{w_{rX}+1}), \tag{7}$$

which can be computable in probabilistic polynomial time. Let $\underline{b}$ be a solution of (7) and $\underline{k}$ the maximal element of Λ_{rX}. Then the same method as RA is applicable as follows:

$$\begin{aligned} M &:= F(\underline{b}/d)d^{w_{rX}} \\ &= \tilde{m}(\underline{b}/d)d^{w_{rX}} + r(\underline{b}/d)X(\underline{b}/d)d^{w_{rX}} \\ &= \tilde{m}(\underline{b}/d)d^{w_{rX}} + r(\underline{b}/d)d^{w_r}X(\underline{b}/d)d^{w_X}, \\ m_{\underline{k}} &= (M\underline{b}^{-\underline{k}})^{e'} (\mathrm{mod}\ d), \\ \tilde{m}_{\underline{k}} &= m_{\underline{k}}^e (\mathrm{mod}\ Nd). \end{aligned}$$

Similarly, we can compute the other coefficients of m. However, using cipher polynomials of the form

$$F_i := \tilde{m} + s_i f + r_i X (i = 1, 2, 3),$$

we may avoid this weakness because $s_i f$ obstructs to get $\tilde{m}(\underline{b}/d)d^{w_X} (\mathrm{mod}\ d)$.

4.7 Ideal decomposition attack

By using the resultant as in section 2.6.4, it is also possible in our case to reconstruct the ideals $I := (\tilde{m}, f, X) \subset \mathbb{Z}[\underline{x}]$, $J := (\tilde{m} + z, f, X) \subset \mathbb{Q}[\underline{x}, z]$ or $\overline{J}^{(\ell)} := \left(\overline{\tilde{m}}^{(\ell)} + z, \overline{f}^{(\ell)}, \overline{X}^{(\ell)}\right) \subset (\mathbb{Z}/\ell\mathbb{Z})[\underline{x}, z]$ from the data (F_1, F_2, X), where z is a new variable and ℓ is a prime number. If one can get $\tilde{m}$, then one can get m. A simple method to avoid this attack is to let $\Lambda_{\tilde{m}} = \Lambda_f = \Lambda_X$ and the coefficients of $\tilde{m}$ be larger than $H(X)$. Then $\tilde{m}$ cannot be determined uniquely because $\tilde{m}' + z \in J$ implies $\tilde{m}' + z + sX + tf \in J$ for any $s, t \in \mathbb{Z}$ (note that $\Lambda_{\tilde{m}} = \Lambda_f = \Lambda_X$). However, in general, we cannot determine $\tilde{m}$ from $\tilde{m}(\underline{a}/d)d^{w_X}$ uniquely even if we know the secret key $\underline{a}$. This reason is as follows: for any $t \in \mathbb{Z}$, $\tilde{m}(\underline{x})$ and $\tilde{m}(\underline{x}) + tX(\underline{x})$ have the same value at $\underline{a}/d$. So, we use modular exponentiation to transform m into $\tilde{m}$ and use

Euler's theorem as in the RSA cryptosystem to recover m from $\tilde{m}(\underline{a}/d)d^{w_X}$ in RA. This is the main idea to avoid this attack.

Now, we analyze the effectiveness of the ideal decomposition attack in detail. We analyze only the Level 2 and the Level 3 attacks because, experimentally, the Level 1 attack is not efficient. First, we analyze the effectiveness of the ideal decomposition attack of Level 2 (see [12], section 3.2), which uses the ideal decomposition

$$(F_1 - F_2, X) = ((s_1 - s_2)f, X) = I_1 \cap I_2 \subset \mathbb{Q}[\underline{x}], \quad (f, X) \subset I_1,$$

to reconstruct from the data (F_1, F_2, X) an ideal $J \subset \mathbb{Q}[\underline{x}, z]$ which coincides with $(\tilde{m} + z, f, X)$. To get $\tilde{m}$, we use the fact that if a Gröbner basis of J is computed, then $\tilde{m}' + z \in J$ if and only if $NF_J(\tilde{m}' + z) = 0$ (see section 2.6.4 for more detail). But, if $\tilde{m}' + z \in J$, then for any integers s and t, $\tilde{m}' + z + sX + tf \in J$ is also satisfied. If the number of choices of the pairs $(s, t) \in \mathbb{Z}^2$ is larger than 2^{128}, we may avoid this attack. All coefficients of $\tilde{m}$ and f are smaller than Nd, but in many cases they are as large as Nd, if $m_{\underline{i}}^e > Nd$. So the possible choices of t may be only 0, 1 or 2. But, if $Nd > 2^{128}H(X)$, the number of the possible choices of s may be larger than 2^{128}. So N should be chosen so that $Nd > 2^{128}H(X)$ and e should be so large that $m_{\underline{i}}^e \geq 2^e > Nd$ for $\underline{i} \in \Lambda_m$. In this case, this attack is not assumed to be effective. Note that, because the absolute value of coefficients of f are as large as those of $\tilde{m}$, the above argument implies that choosing N satisfying $Nd > 2^{128}H(X)$ may complicate finding f from the ideal J or I_1.

Next, we analyze the effectiveness of the ideal decomposition attack of Level 3 (see [12], section 3.3). We assume that d is a prime number. We note that if one got $\overline{\tilde{m}}^{(d)}$, then one can get m. So one does not need to get $\tilde{m}$. It is possible to reconstruct an ideal $\bar{J}^{(d)} \subset (\mathbb{Z}/d\mathbb{Z})[\underline{x}, z]$ which coincides with $\left(\overline{\tilde{m}}^{(d)} + z, \bar{f}^{(d)}, \overline{X}^{(d)}\right)$ from tha data (F_1, F_2, X) (see the algorithm in 2.6.4). Let $\tilde{m}'(\underline{x}) := \sum_{\underline{i} \in \Lambda_{\tilde{m}}} \tilde{m}'_{\underline{i}} \underline{x}^{\underline{i}}$, where $\tilde{m}'_{\underline{i}}$ are variables for $\underline{i} \in \Lambda_{\tilde{m}}$. Assume that a Gröbner basis of $\bar{J}^{(d)}$ is computed. Let J be the ideal of $(\mathbb{Z}/d\mathbb{Z})[m'_{\underline{c}0}, \cdots, \tilde{m}'_{\underline{k}}]$ generated by the coefficients of $NF_{\bar{J}^{(d)}}(\tilde{m}' + z)$. Let $\{g_1, \cdots, g_h\}$ be a Gröbner basis of J. Then g_i is linear with respect to its variables for each $1 \leq i \leq h$. So we can use linear algebra techniques to solve $NF_{\bar{J}^{(d)}}(\tilde{m}' + z) = 0$. Let A be the coefficient matrix of the equation system $g_1 = \cdots = g_h = 0$. Let D be the dimension of the kernel of the linear map $\mathbb{F}_d^{\#\Lambda_{\tilde{m}}} \to \mathbb{F}_d^h$ defined by A. Then the number of polynomials in $\bar{J}^{(d)}$ having the same form as $\overline{\tilde{m}}^{(d)} + z$ is d^D. So if $d^D > 2^{128}$, the Level 3 attack is not effective. Experimentally, D is at least 2. Thus, this attack is not assumed to be effective if $d^2 \geq 2^{128}$ $\left(d \geq 2^{64}\right)$.

Next, we assume that $d = \prod_{1 \leq i \leq k} p_i$ ($k \geq 2$ and p_i are distinct prime numbers for $1 \leq i \leq k$). If one got $\overline{\tilde{m}}^{(p_i)}$ for $1 \leq i \leq k$, then one can get $\overline{\tilde{m}}^{(d)}$ and m by the Chinese Remainder Theorem. However, because of the above argument we may also avoid this attack, if d is sufficiently large, for example $d^2 > 2^{128}$. Note that if $d = \prod_{1 \leq i \leq k} p_i^{e_i}$ and $e_i \geq 2$ for some i, this attack may not be directly applicable, because $\mathbb{Z}/p^{e_i}\mathbb{Z}$ is not a domain if $e_i \geq 2$. But, it is possible to lift a polynomial $\overline{\tilde{m}}^{(p_i)} \in (\mathbb{Z}/p_i\mathbb{Z})[\underline{x}]$ to a polynomial $\overline{\tilde{m}}^{(p_i^{e_i})} \in (\mathbb{Z}/p_i^{e_i}\mathbb{Z})[\underline{x}]$ for $1 \leq i \leq n$. There are $p_i^{e_i - 1}$ ways of such a lifting. So we may also avoid this attack, if d is sufficiently large, for example $d \geq 2^{64}$.

5 Sizes of keys and cipher polynomials

In this section we estimate the sizes of keys and cipher polynomials so that our cryptosystem can be expected to have 128 bit-security. First, we estimate the size of a secret key and a public key. A typical brute force attack is as follows: One chooses a random vector $(b_1, \ldots, b_{n-1})$ and factorize the polynomial $X\left(\frac{b_1}{d}, \ldots, \frac{b_{n-1}}{d}, x_n\right)$ in x_n. If $X\left(\frac{b_1}{d}, \ldots, \frac{b_{n-1}}{d}, x_n\right)$ has a factor of the form $\left(x_n - \frac{b_n}{d}\right)$ for some integer b_n, then $\left(\frac{b_1}{d}, \ldots, \frac{b_n}{d}\right)$ is a solution to $X = 0$. If $\gcd\left(\prod_i b_i, d\right) = 1$, then using the solution $\left(\frac{b_1}{d}, \ldots, \frac{b_n}{d}\right)$, one can get m by taking the same steps as the decryption process. So we should choose a secret key $\underline{a} = (a_1, \ldots, a_n)$ such that $|a_i|$ is sufficiently large for $i = 1, \ldots, n$ to avoid the brute force attack. Since the probability that a random integer b is prime to d is $\frac{\varphi(d)}{d}$ ($\varphi(\cdot)$ is the Euler's function), the number of choices of the vector $(b_1, \ldots, b_{n-1})$ which satisfies $\frac{2^{\lceil \frac{128}{n-1} \rceil} d}{\varphi(d)} \leq |b_i| < \frac{2^{\lceil \frac{128}{n-1} \rceil + 1} d}{\varphi(d)}$ and $\gcd\left(\prod_i b_i, d\right) = 1$ is at least $2^{\left\lceil \frac{128}{n-1} \right\rceil (n-1)} \geq 2^{128}$. Thus we should choose a secret key so that

$$\frac{2^{\left\lceil \frac{128}{n-1} \right\rceil} d}{\varphi(d)} \leq |a_i| < \frac{2^{\left\lceil \frac{128}{n-1} \right\rceil + 1} d}{\varphi(d)} \tag{8}$$

for $i = 1, \ldots, n$. We assume (8). Let $\underline{k}$ be the maximal element of Λ_X and Λ'_X be as in section 3.4.1. We assume that X is constructed by the method described in section 3.4.1. There are infinitely many solutions of (4). We claim that we can choose a solution $(c_{\underline{0}}, c_{\underline{k}})$ such that $|c_{\underline{0}}| \leq |\underline{a}^{\underline{k}}|$ and $|c_{\underline{k}}| \leq d^{w_X}$, if the following inequality is satisfied:

$$\left|\underline{a}^{\underline{k}} d^{w_X}\right| > \left|\sum_{\underline{i} \in \Lambda'_X} c_{\underline{i}} \underline{a}^{\underline{i}} d^{w_X - \sum \underline{i}}\right|. \tag{9}$$

To see this, let $A := \left|\sum_{\underline{i}\in\Lambda'_X} c_{\underline{i}}\underline{a}^{\underline{i}}d^{w_X-\sum\underline{i}}\right|$. If (x_0, y_0) is a solution to

$$\left|\underline{a}^{\underline{k}}\right| x + d^{w_X} y = A,$$

then all solutions are given by $\left(x_0 + kd^{w_X}, y_0 - k\underline{a}^{\underline{k}}\right)$ for $k \in \mathbb{Z}$. Looking at the first lattice point (x, y) on the line $|\underline{a}^{\underline{k}}|x + d^{w_X}y = A$ with $x > 0$, we find a solution (x, y) such that $x \le d^{w_X}$ and $y \le |\underline{a}^{\underline{k}}|$. Thus, we have proved the above claim.

In many cases the minimum size of the solutions of (4) satisfies $|c_{\underline{0}}| \approx |\underline{a}^{\underline{k}}|$ and $|c_{\underline{k}}| \approx d^{w_X}$. If the $|c_{\underline{i}}|$'s are so small that (9) is satisfied, then we may assume that

$$H(X) = \begin{cases} |c_{\underline{0}}| \approx |\underline{a}^{\underline{k}}| < \left(\frac{2^{(\lceil\frac{128}{n-1}\rceil+1)}d}{\varphi(d)}\right)^{w_X} & \text{if } |\underline{a}^{\underline{k}}| \gg d^{w_X}, \\ |c_{\underline{k}}| \approx d^{w_X} & \text{if } |\underline{a}^{\underline{k}}| \ll d^{w_X}. \end{cases}$$

On the other hand, as mentioned in section 4.7, N, d and e should be chosen so that $Nd > 2^{128}H(X)$, $d \ge 2^{64}$ and $2^e > Nd$, respectively. We must determine an upper bound of Nd and d to estimate the size of e and $c_{\underline{k}}$, respectively. We assume that $H(X) = c_{\underline{k}}$, $2^{64} \le d < 2^{65}$ and $2^{128}H(X) \le 2^{128}d^{w_X} < 2^{128+65w_X} \le Nd$. Then $c_{\underline{k}} \le 2^{65w_X}$ and $N \ge 2^{128+65(w_X-1)}$. If we assume that $2^{128+65(w_X-1)} \le N < 2^{128+65(w_X-1)+1} = 2^{129+65(w_X-1)}$, then we should choose e so that $e \ge 129 + 65w_X$ because $Nd < 2^{129+65w_X}$. It remains to estimate the size of $|c_{\underline{i}}|$ for $\underline{i} \in \Lambda'_X$. We think that the size of these coefficients may be small enough to keep the size of the public key reasonable even though we cannot prove it. For example, if $|c_{\underline{i}}| < 2^{10}$, then the size of X, that is $\sum_{\underline{i}\in\Lambda_X}$(bit length of $c_{\underline{i}}$), is at most $\left(\lceil\frac{128}{n-1}\rceil + 1 + \lceil\log_2 d - \log_2\varphi(d)\rceil\right)w_X + 65w_X + 10(\#\Lambda_X - 2) = \left(\lceil\frac{128}{n-1}\rceil + 66 + \lceil\log_2 d - \log_2\varphi(d)\rceil\right)w_X + 10\#\Lambda'_X$ bits under the above assumptions. If $w_X \approx \#\Lambda_X = \Lambda'_X + 2$, then the size of $X \approx (\lceil\frac{128}{n-1}\rceil + 76 + \lceil\log_2 d - \log_2\varphi(d)\rceil)w_X$ bits. Then the size of the secret key and the public key is at most $\left(\lceil\frac{128}{n-1}\rceil + 1\right)n + \lceil\log_2 d - \log_2\varphi(d)\rceil$ bits and $\left(\lceil\frac{128}{n-1}\rceil + 76 + \lceil\log_2 d - \log_2\varphi(d)\rceil\right)w_X + 65 + \lceil\log_2 e\rceil$ bits, respectively.

Next, we estimate the size of F_i for $i = 1, 2, 3$. We may assume that the size of F_i is about the same as that of $2s_if$ because $\Gamma_f = \Gamma_{r_i}$ and $\Gamma_{s_i} = \Gamma_X$. Since $\Lambda_X = \Lambda_f = \Lambda_{s_i}$, $\#\Lambda_X \le w_X$, $H(f) < Nd < 2^{129+65w_X}$ and $H(s_i) \approx H(X) < 2^{65w_X}$, we have

$$H(s_if) \le \#\Lambda_X H(f)H(s_i) < 2^{129+130w_X}w_X.$$

It implies that the size of $2s_if$ is at most $(130 + 130w_X + \lceil\log_2 w_X\rceil)\#\Lambda_{s_if}$ bits. So, it is important to estimate $\#\Lambda_{s_if}$,

Table 4 Size of keys of our cryptosystem

No.	n	w_X	$\#\Lambda_X$	Secret key (bit)	Public key (bit)
1	3	5	4	198	739
2	3	5	5	198	747
3	3	7	4	198	1000
4	3	7	7	198	1031
5	3	10	4	198	1393
6	3	10	7	198	1420
7	3	10	10	198	1450

explicitly. We assume $\Lambda_f = \Lambda_{s_i} = \left\{\underline{k}_1, \ldots, \underline{k}_{\#\Lambda_f}\right\}$. Then we can write

$$\begin{aligned} s_if &= \left(\sum_{\underline{j}\in\Lambda_{s_i}} s^{(i)}_{\underline{j}}\underline{x}^{\underline{j}}\right)\left(\sum_{\underline{j}\in\Lambda_f} f_{\underline{j}}\underline{x}^{\underline{j}}\right) \\ &= \sum_{\underline{j}} s^{(i)}_{\underline{k}_j}f_{-\underline{k}_j}\underline{x}^{2\underline{k}_j} + \sum_{\underline{j}\ne\underline{h}}\left(s^{(i)}_{\underline{k}_j}f_{-\underline{k}_h} + s^{(i)}_{\underline{k}_h}f_{-\underline{k}_j}\right)\underline{x}^{\underline{k}_j+\underline{k}_h}. \end{aligned}$$

It implies that

$$\#\Lambda_{s_if} \le \frac{\#\Lambda_f^2 - \#\Lambda_f}{2} + \#\Lambda_f \le \frac{w_X^2 - w_X}{2} + w_X.$$

Thus, the size of $2s_if$ is at most

$$\begin{aligned} &\left(\frac{w_X^2 - w_X}{2} + w_X\right)(130 + 130w_X + \lceil\log_2 w_X\rceil) \\ &\quad = \frac{1}{2}\left(w_X^2 + w_X\right)(129 + 130w_X + \lceil\log_2 w_X\rceil) \end{aligned}$$

bits. Since $2^{128+65(w_X-1)} \le N < 2^{129+65(w_X-1)}$, we conclude that the size of ciphertext is at most

$$\frac{3}{2}\left(w_X^2 + w_X\right)(129 + 130w_X + \lceil\log_2 w_X\rceil) + 129 + 65(w_X - 1)$$

bits.

6 Examples

In Table 4 and Table 5 we give examples of the size of keys and ciphertexts. In Table 6 we also give examples of the time which it took to encrypt and decrypt.

Table 5 Size of ciphertext of our cryptosystem

No.	n	w_X	$\#\Lambda_X$	F_1(bit)	F_2(bit)	F_3(bit)	N(bit)
1	3	5	4	7442	7443	7440	387
2	3	5	5	10755	10748	10752	390
3	3	7	4	9946	9942	9947	521
4	3	7	7	23907	23915	23917	515
5	3	10	4	13685	13684	13688	717
6	3	10	7	33658	33659	33667	717
7	3	10	10	57740	57749	57767	719

Table 6 Encryption time and decryption time

No.	n	w_X	$\#\Lambda_X$	enc. time (ms)	dec. time (ms)
1	3	5	4	39	34
2	3	5	5	38	33
3	3	7	4	38	34
4	3	7	7	38	34
5	3	10	4	39	34
6	3	10	7	39	36
7	3	10	7	40	40

We use a computer with 2.80 GHz CPU (Intel(R) Core(TM) i7-3840QM) and 8GB memory. The OS is Windows 8.1 Pro 64 bit. We implemented in Magma V2.19-7 [5] and the source code of our cryptosystem (file name: `crypto-okumura.txt`) is available at `http://imi.kyushu-u.ac.jp/~s-okumura/`.

7 Conclusion

In this paper we have proposed a new public key cryptosystem based on diophantine equations and analyzed its security. It is a number field analogue of the ASC, incorporating a key idea, to avoid some attacks, of "twisting" the plaintext by using some modular arithmetic and Euler's theorem as in the RSA cryptosystem. Another key idea is to use a polynomial, as the public key, of degree increasing type to recover the plaintext. In this paper we have not studied the hardness of solving diophantine equations of degree increasing type. Investigating the security of our cryptosystem by using this special type of diophantine equations is a future work.

Endnotes

[a]The size of a_i should be $|a_i| \geq \frac{2^{\lceil \frac{128}{n-1} \rceil + 1} d}{\varphi(d)}$ for $i = 1, \ldots, n$, where $\varphi(\cdot)$ is the Euler function and d is an integer which we will choose below. (For the reason of this choice, see section 5).

[b]The sizes of d and e should be $d \geq 2^{64}$ and $e \geq 129 + 65w$, respectively. (For the reason of this choice, see section 5).

Acknowledgements

I am grateful to my supervisor Yuichiro Taguchi for comments, corrections, and suggestions on this research. I am also grateful to Koichiro Akiyama, Noriko Hirata-Kohno, Attila Pethő, Takakazu Satoh and Tsuyoshi Takagi for useful comments, suggestions and discussions.

Received: 23 December 2014 Revised: 30 March 2015
Accepted: 26 April 2015
Published online: 05 June 2015

References

1. Akiyama, K., Goto, Y., Miyake, H.: An algebraic surface cryptosystem. In: Proceedings of PKC'09, Lecture Notes in Comput. Sci., vol. 5443, pp. 425–442. Springer, Berlin Heidelberg, (2009)
2. Berlekamp, E.R: Factoring polynomials over large finite fields. Math. Comput. **24**, 713–735 (1970)
3. Beukers, F., Tengely, S.: An implementation of Runge's method for Diophantine equations, (2005). available at arXiv:math/0512418
4. Bilu, Y.: Effective analysis of integral points on algebraic curves. Israel J. Math. **90**, 235–252 (1995)
5. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. J. Symbolic Comput. **24**, 235–265 (1997)
6. Bugeaud, Y., Mignotte, S., Siksek, S., Stoll, M., Tengely, S.: Integral points on hyperelliptic curves. Algebra Number Theory. **2**, 859–885 (2008)
7. Cantor, D.G, Zassenhaus, H.: On Algorithms for Factoring Polynomials over Finite Fields. Math. of Computation. **36**, 587–592 (1981)
8. Cox, D., Little, J., O'Shea, D.: Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra, 3rd., Undergraduate Texts in Mathematics. Springer Verlag, New York (2007)
9. Cusick, T.W: Cryptoanalysis of a public key system based on diophantine equations. Inform. Process. Lett. **56**, 73–75 (1995)
10. Davis, M., Matijasevič, Y., Robinson, J.: Hilbert's tenth problem, Diophantine equations: positive aspects of a negative solution, In: Browder, FE (ed.) Mathematical developments arising from hilbert problems (Proc. Sympos. Pure Math., Vol. XXVIII, Northern Illinois Univ., De Kalb, Ill., 1974), pp. 323–378. (loose erratum) Amer. Math. Soc., Providence, R. I., 1976
11. Diffie, W., Hellman, M.: New direction in cryptography. Trans. Inf. Theory. **22**, 644–654 (1976)
12. Faugére, J.C, Spaenlehauer, P.-J.: Algebraic Cryptanalysis of the PKC'2009 Algebraic Surface Cryptosystem. In: Proceedings of PKC'10, Lecture Notes in Comput. Sci., vol. 6056, pp. 35–52. Springer, Berlin Heidelberg, (2010)
13. Győry, K.: Solving Diophantine equations by Baker's theory. In: A panorama of number theory of the view from Baker's garden (Zürich, 1999), pp. 38–72. Cambridge University Press, Cambridge, England, (2002)
14. Hindry, M., Silverman, J.H: Diophantine geometry: an introduction, Graduate Texts in Mathematics, 201. Springer, New York (2000)
15. Hirata-Kohno, N., Pethő, A.: On a key exchange protocol based on Diophantine equations. Infocommunications J. **16**(2), 168–184 (1987)
16. Iwami, M.: A Reduction Attack on Algebraic Surface Public-Key Cryptosystems. In: Kapur, D (ed.) ASCM 2007. LNCS, vol. 5081, pp. 323–332. Springer, Heidelberg, (2008)
17. Koblitz, N.: Elliptic curve cryptosystems. Math. Comput. **48**, 203–209 (1987)
18. Lenstra, A.K, Lenstra, H.W, (ed.): The Development of the Number Field Sieve, Lecture Notes in Mathematics, vol. 1554. Springer-Verlag, Berlin Heidelberg (1993)
19. Lin, C.H, Chang, C.C, Lee, R.CT: A new public-key cipher system based upon the diophantine equations. IEEE Trans. Comp. **44**, 13–19 (1995)
20. Manders, K., Adleman, L.: NP-complete decision problems for binary quadratics. J. Comput. Syst. Sci. **24**, 713–735 (1970)
21. Mason, R.C: Diophantine Equations over Function Fields, London Mathematical Society Lecture Note Series, vol. 96. Cambridge University Press, Cambridge, England (1984)
22. Miller, V.S: Use of elliptic curves in cryptography. Abstracts for Crypto. '85. Lect. Notes Comput. Sci. **218**, 417–426 (1986)
23. Mochizuki, S.: Inter-universal Teichmüller Theory I: Construction of Hodge Theaters, II: Hodge-Arakelov-theoretic Evaluation, II: Canonical Splittings of the Log-theta-lattice, IV: Log-volume Computations and Set-theoretic Foundations. available at http://www.kurims.kyoto-u.ac.jp/~motizuki/papers-english.html
24. Ogura, N.: On Multivariate Public-key cryptosystems. PhD thesis, Tokyo Metropolitan University (2012)
25. Pheidas, T.: Hilbert's tenth problem for fields of rational functions over finite fields. Invent. Math. **103**(1), 1–8 (1991)
26. Poulakis, D., Voskos, E.: On the practical solution of genus zero Diophantine equations. J. Symbolic Comput. **30**, 573–582 (2000)
27. Rivest, R.L, Shamir, A., Adleman, L.: A method for obtaining digital signatures and public key cryptosystems. Commun. ACM. **21**, 120–126 (1987)
28. Shor, P.: Algorithms for Quantum Computation: Discrete Logarithm and Factoring. In: Proc. 35th Annual Symposium on Foundations of Computer Science, pp. 124–134 (1994)
29. Stothers, W. W.: Polynomial identities and hauptmoduln. Quart. J. Math. Oxford Ser. (2). **32**(127), 349–370 (1981)

30. Stroeker, R.J, Tzanakis, N.: Computing all integer solutions of a genus 1 equation. Math. Comput. **72**, 1917–1933 (2003)
31. Uchiyama, S., Tokunaga, H.: On the Security of the Algebraic Surface Public-key Cryptosystems (in Japanese). In: Proceedings of of SCIS 2007, CD-ROM 2C1-2, (2009)
32. Voloch, F.: Breaking the Akiyama-Goto algebraic surface cryptosystem. Arithmetic, Geometry, Cryptography and Coding Theory, CIRM meeting (2007)
33. Weil, A.: Sur les courbes algébriques et les variétés qui s'en déduisent. Actualités Sci. Ind., no. 1041; Publ. Inst. Math. Univ. Strasbourg **7** (1945). Hermann, Paris, 1948. iv+85 pp.
34. Yosh, H.: The key exchange cryptosystem used with higher order Diophantine equations. Int. J. Netw. Secur. Appl. **3**, 43–50 (2011)