Breaking Pairing-Based Cryptosystems Using η_T Pairing over $GF(3^{97})$[*]

Takuya Hayashi[1,**], Takeshi Shimoyama[2], Naoyuki Shinohara[3], and Tsuyoshi Takagi[1]

[1] Kyushu University
[2] FUJITSU LABORATORIES Ltd.
[3] National Institute of Information and Communications Technology

Abstract. In this paper, we discuss solving the DLP over $GF(3^{6\cdot97})$ by using the function field sieve (FFS) for breaking paring-based cryptosystems using the η_T pairing over $GF(3^{97})$. The extension degree 97 has been intensively used in benchmarking tests for the implementation of the η_T pairing, and the order (923-bit) of $GF(3^{6\cdot97})$ is substantially larger than the previous world record (676-bit) of solving the DLP by using the FFS. We implemented the FFS for the medium prime case, and proposed several improvements of the FFS. Finally, we succeeded in solving the DLP over $GF(3^{6\cdot97})$. The entire computational time requires about 148.2 days using 252 CPU cores.

Keywords: pairing-based cryptosystems, η_T pairing, discrete logarithm problems, function filed sieve.

1 Introduction

After the advent of the tripartite Diffie-Hellman (DH) key exchange scheme [21] and ID-based encryption using pairing [11], plenty of attractive pairing-based cryptosystems have been proposed, for example, short signature [13], keyword searchable encryption [10], efficient broadcast encryption [12], attribute-based encryption [30], and functional encryption [28]. Pairing-based cryptosystems have become a major research topic in cryptography.

Pairing-based cryptosystems are constructed on the groups G_1, G_1' and G_2 of the same order with a bilinear pairing $G_1 \times G_1' \to G_2$. The security of pairing-based cryptosystems is based on the difficulty in solving several number-theoretic problems such as the computational/decisional bilinear DH problem (CBDH/DBDH), strong DH problem (SDH), decisional linear problem (DLIN), and symmetric external DH problem (SXDH). However, the most important number-theoretic problem in pairing-based cryptosystems is the discrete logarithm problem (DLP) on G_1, G_1', and G_2. All the other number-theoretic problems above are no longer intractable once the DLP on G_1, G_1', or G_2 is broken. Therefore, it is important to investigate the difficulty in solving the DLP.

[*] The full-version of this paper is appeared in [19].

[**] The author is supported by JSPS KAKENHI Grant Number 10J56502.

Table 1. Summary of time data for solving DLP over $GF(3^{6\cdot 97})$

phase	method	time	machine environment
collecting relations	lattice sieve	53.1 days	212 CPU cores
linear algebra	parallel Lanczos	80.1 days	252 CPU cores
individual logarithm	rationalization and special-Q descent	15.0 days	168 CPU cores
total		148.2 days	252 CPU cores

One of the most efficient algorithms for implementing the pairing is the η_T pairing [5] defined over a supersingular elliptic curve E on the finite field $GF(3^n)$, where n is a positive integer. Since the embedding degree of E is 6, the η_T pairing can reduce a DLP over E on $GF(3^n)$, which is called an ECDLP, to a DLP over $GF(3^{6n})$. Joux proposed the (probably) first cryptographic scheme [21] that uses the pairing over E. Boneh *et al.* then applied the pairing over E to the short signature scheme [13], where a point (x, y) on E for extension degree $n = 97$ can be represented as a signature value, e.g., $x =$ `KrpIcV0O9CJ8iyBS8MyVkNrMyE`. At CRYPTO 2002, Barreto *et al.* presented algorithms for efficiently computing Tate pairing over E [6]. Many high-speed implementations of pairing over E have subsequently been proposed [3, 7–9, 17, 18, 25]. For many of these implementations, benchmark tests using the extension degree $n = 97$ have been conducted. Therefore, we focus on the DLP over finite field $GF(3^{6\cdot 97})$ in this paper. The cardinality of the subgroup of the supersingular elliptic curve is 151 bits, and that of $GF(3^{6\cdot 97})$ is 923 bits. The size of our target DLP is 247 bits larger than the previous world record of solving the DLP over $GF(3^{6\cdot 71})$, whose cardinality is 676 bits [20]. The current world record for solving an ECDLP is the 112-bit ECDLP [14]. Pollard's ρ method is used for solving the 112-bit ECDLP, and has not reached the ability for solving the 151-bit ECDLP over the subgroup of E.

In this paper, we analyze the difficulty in solving the DLP over $GF(3^{6\cdot 97})$ by using the function field sieve (FFS), which is known as the asymptotically fastest algorithm [1, 2]. Since the FFS proposed by Joux and Lercier (JL06-FFS) [24] is suitable for solving the DLP over a finite field whose characteristic is small, we use the JL06-FFS and propose several efficient techniques for increasing its speed. Note that the FFS generally consists of four phases: polynomial selection, collecting relations, linear algebra, and individual logarithm, and the time-consuming phases are collecting relations and linear algebra. For the collecting relations phase, we applied several techniques; lattice sieve for the JL06-FFS, lattice sieve with single instruction multiple data (SIMD), and optimization for our parameters. These techniques enable the sieving program to run about 6 times faster. In the linear algebra phase, we applied careful treatments of singleton-clique and merging [15] to the Galois action originating from extension degree 6 of $GF(3^{6\cdot 97})$, with which the size of the matrix used for the Lanczos method is reduced to approximately 30%. By implementing the JL06-FFS with our improvements, we succeeded in solving the DLP over $GF(3^{6\cdot 97})$ by using 252 CPU cores (Core2 quad, Xeon, etc) for the target problem discussed in Section 3.1. As shown in Table 1, the computations required 53.1 days for the collecting relations phase, 80.1 days for the linear algebra phase, and 15.0 days for the individual logarithm phase. Thus, a total of 148.2 days were required to solve the DLP over $GF(3^{6\cdot 97})$ by using 252 CPU cores. Our computational results contribute to the secure use of pairing-based cryptosystems with the η_T pairing.

2 Pairing-Based Cryptosystems and Discrete Logarithm Problem (DLP)

In this section, we briefly explain the security of pairing-based cryptosystems and give a general overview of the function field sieve (FFS). We also mention its parameters such as the smoothness bound B.

2.1 Pairing-Based Cryptosystems and DLP

Many efficient cryptographic protocols using a bilinear pairing have been proposed (for example [10–13, 21, 28]), and high-speed implementations for the η_T pairing have been reported (for example [3, 6–9, 17, 18, 25]). We discuss the security of pairing-based cryptosystems with the η_T paring over $GF(3^n)$ for an integer n. The security of pairing-based cryptosystems with the η_T paring depends on the difficulty in solving the DLP over the supersingular elliptic curves. Additionally, MOV reduction [27] reduces this problem to a DLP over $GF(3^{6n})^*$ since the embedding degree of the η_T pairing is 6.

In particular, the η_T pairing is a bilinear map such that $\eta_T : G_1 \times G_1 \to G_2$, where G_1 is an additive subgroup of a supersingular elliptic curve over $GF(3^n)$, G_2 is a cyclic subgroup of $GF(3^{6n})^*$, and the cardinalities of G_1, G_2 are the same prime number P. The security of pairing-based cryptosystems with the η_T pairing depends on the difficulty of not only an ECDLP over G_1 but also a DLP over G_2 by MOV reduction. To explain this fact, we take ID-based encryption constructed on pairing-based cryptosystems as an example. The ID-based encryption has a master key $s_{key} \in \mathbb{Z}_P$. Each user ID is deterministically transformed into a point $\mathcal{Q}_{ID} \in G_1$, and the secret key $\mathcal{S}_{ID}$ is defined by $[s_{key}]\mathcal{Q}_{ID}$. Therefore, solving the ECDLP over G_1, namely $\mathcal{S}_{ID} = [s_{key}]\mathcal{Q}_{ID}$, we obtain the master key $s_{key} = \log_{\mathcal{Q}_{ID}} \mathcal{S}_{ID}$. Additionally, for an arbitrary point $\mathcal{R} \in G_1$, we compute $\eta_T(\mathcal{S}_{ID}, \mathcal{R}), \eta_T(\mathcal{Q}_{ID}, \mathcal{R}) \in G_2$, and then have $\eta_T(\mathcal{S}_{ID}, \mathcal{R}) = \eta_T([s_{key}]\mathcal{Q}_{ID}, \mathcal{R}) = \eta_T(\mathcal{Q}_{ID}, \mathcal{R})^{s_{key}} \in G_2$. This implies that $s_{key} = \log_{\eta_T(\mathcal{Q}_{ID},\mathcal{R})} \eta_T(\mathcal{S}_{ID}, \mathcal{R})$ is also available by solving the DLP over G_2. In this paper, we discuss the DLP over a subgroup of $GF(3^{6n})^*$.

2.2 General Overview of FFS

The FFS is the asymptotically fastest algorithm for solving a DLP over finite fields of small characteristics. Adleman proposed the first FFS in 1994 [1]. After that, several variants of the FFS have been proposed; Adleman and Huang improved the FFS [2], and Joux and Lercier proposed two more practical FFS's, JL02-FFS [23] and JL06-FFS [24]. The details of JL06-FFS are explained in Sections 3.2.

In this section, we give a general overview of an FFS that consists of four phases: polynomial selection, collecting relations, linear algebra, and individual logarithm. In the overview, we aim at computing $\log_g T$ where $T \in \langle g \rangle \subset GF(3^{6n})^*$.

Polynomial Selection Phase: We select κ from $\kappa = 1, 2, 3, 6$ for the coefficient field of $GF(3^\kappa)[x]$, and a bivariate polynomial $H(x, y) \in GF(3^\kappa)[x, y]$ such that H satisfies the eight conditions proposed by Adleman [1] and $\deg_y H = d_H$ for a given parameter value d_H. We compute a random polynomial $m \in GF(3^\kappa)[x]$ of degree d_m and a monic irreducible polynomial $f \in GF(3^\kappa)[x]$ such that

$$H(x, m) \equiv 0 \pmod{f}, \;\; \deg f = 6n/\kappa. \tag{1}$$

We then have $GF(3^{6n}) \cong GF(3^\kappa)[x]/(f)$. Moreover, there is a surjective homomorphism

$$\xi : \begin{cases} GF(3^\kappa)[x,\, y]/(H) & \rightarrow & GF(3^{6n}) \cong GF(3^\kappa)[x]/(f) \\ \qquad y & \mapsto & m. \end{cases}$$

We select a positive integer B as a smoothness bound, and define a rational factor base $F_R(B)$ and an algebraic factor base $F_A(B)$ as follows.

$$F_R(B) = \{\mathfrak{p} \in GF(3^\kappa)[x] \mid \deg(\mathfrak{p}) \le B,\, \mathfrak{p} \text{ is monic irreducible}\}, \tag{2}$$

$$\begin{aligned} F_A(B) = \{\langle \mathfrak{p}, y - t\rangle \in \operatorname{Div}(GF(3^\kappa)[x, y]/(H)) \mid \\ \mathfrak{p} \in F_R(B),\, H(x, t) \equiv 0 \bmod \mathfrak{p}\}, \end{aligned} \tag{3}$$

where $\operatorname{Div}(GF(3^\kappa)[x,\, y]/(H))$ is the divisor group of $GF(3^\kappa)[x,\, y]/(H)$ and $\langle \mathfrak{p},\, y - t\rangle$ is a divisor generated by $\mathfrak{p}$ and $y - t$. Note that $F_R(0) = F_A(0) = \{\emptyset\}$. We simply call the set $F_R(B) \cup F_A(B)$ a factor base and the set $F_R(k)\backslash F_R(k - 1) \cup F_A(k)\backslash F_A(k - 1)$ a factor base of degree k for $k = 1, 2, \ldots, B$.

Collecting Relations Phase: We select positive integers R, S and collect a sufficient amount of pairs $(r, s) \in (GF(3^\kappa)[x])^2$ such that

$$\deg r \le R,\, \deg s \le S,\, \gcd(r, s) = 1, \tag{4}$$

$$rm + s = \prod_{\mathfrak{p}_i \in F_R(B)} \mathfrak{p}_i^{a_i}, \tag{5}$$

$$\langle ry + s\rangle = \sum_{\langle \mathfrak{p}_j, y - t_j\rangle \in F_A(B)} b_j \langle \mathfrak{p}_j, y - t_j\rangle, \tag{6}$$

for some non-negative integers a_i, b_j by using a sieving algorithm such as the lattice sieve discussed in Section 4.1. To efficiently compute b_j in (6), we use the following equivalent property instead of (6):

$$(-r)^{d_H} H(x,\, -s/r) = \prod_{\langle \mathfrak{p}_j, y - t_j\rangle \in F_A(B)} \mathfrak{p}_j^{b_j}. \tag{7}$$

The (r, s) satisfying (4), (5), and (7) is called a B-smooth pair. Let h be the class number of the quotient field of $GF(3^\kappa)(x)[y]/(H)$ and assume that h is coprime to $(3^{6n} - 1)/(3^\kappa - 1)$. Then the following congruent holds:

$$\sum_{\mathfrak{p}_i \in F_R(B)} a_i \log_g \mathfrak{p}_i \equiv \sum_{\langle \mathfrak{p}_j, y - t_j \rangle \in F_A(B)} b_j \log_g \mathfrak{s}_j \pmod{(3^{6n} - 1)/(3^\kappa - 1)}, \quad (8)$$

where $\mathfrak{s}_j = \xi(\mathfrak{t}_j)^{1/h}$, $\langle \mathfrak{t}_j \rangle = h \langle \mathfrak{p}_j, y - t_j \rangle$. We call the congruent (8) "relation" in this paper. Moreover, free relation [20] provides additional relations without computation with a sieving algorithm.

Linear Algebra Phase: We generate a system of linear equations described as a large matrix from those collected relations and reduce the rank of the matrix by filtering [15]. The reduced system of linear equations is solved using the parallel Lanczos method [4, 20] or other methods, and the discrete logarithms of elements in the factor base are obtained:

$$\log_g \mathfrak{p}_1, ..., \log_g \mathfrak{p}_{\#F_R(B)}, \log_g \mathfrak{s}_1, ..., \log_g \mathfrak{s}_{\#F_A(B)}.$$

Individual Logarithm Phase: Note that our goal is to compute $\log_g T$. Therefore, we find integers a_i, b_j using the special-Q descent [24] such that,

$$\log_g T \equiv \sum_{\mathfrak{p}_i \in F_R(B)} a_i \log_g \mathfrak{p}_i + \sum_{\langle \mathfrak{p}_j, y - t_j \rangle \in F_A(B)} b_j \log_g \mathfrak{s}_j \pmod{(3^{6n} - 1)/(3^\kappa - 1)}.$$

The computational time for the individual logarithm phase is smaller than those for the collecting relations and linear algebra phases.

3 Target Problem for $n = 97$ and Setting of Parameters for FFS

We discuss solving the DLP over a subgroup of $GF(3^{6 \cdot 97})^*$, where the cardinality of the subgroup is 151 bits. To estimate the time complexity of solving such a DLP, we unintentionally set a target problem determined from the circular constant π and natural logarithm e. The details are explained in Section 3.1. To solve the target problem effectively, we select the parameter values of the FFS and estimate important numbers, e.g., the number of elements in the factor base, for it. The details are given in Section 3.2.

3.1 Target Problem

For pairing-based cryptosystems, many high-speed implementations of the η_T pairing over supersingular elliptic curves on $GF(3^n)$ have been reported [3, 6–9, 17, 18, 25], and many benchmark tests using the η_T pairing have been conducted for $GF(3^{97})$. In this paper, we deal with a supersingular elliptic curve defined by

$$E := \{(x, y) \in GF(3^{97})^2 \ : \ y^2 = x^3 - x + 1\} \cup \{\mathcal{O}\},$$

where $\mathcal{O}$ is the point at infinity. The order of the E is $3^{97} + 3^{49} + 1 = 7P_{151}$ where P_{151} is a 151-bit prime number as follows:

$$P_{151} = 2726865189058261010774960798134976187171462721.$$

Next, let G_1 be the subgroup of E of order P_{151} and let G_2 be the subgroup of $GF(3^{6\cdot 97})^*$ of order P_{151}. Note that, since orders of G_1 and G_2 are prime numbers, every element of $G_1 \backslash \{\mathcal{O}\}$ and $G_2 \backslash \{1\}$ is a generator of G_1 and G_2, respectively. The η_T pairing for $n = 97$ is a map from $G_1 \times G_1$ to G_2.

Our goal is to solve the ECDLP in G_1. To set our target problem unintentionally, we select two elements $\mathcal{Q}_\pi, \mathcal{Q}_e$ in G_1, which correspond to the circular constant π and natural logarithm e, respectively. We explain how we select $\mathcal{Q}_\pi$ and $\mathcal{Q}_e$ as follows. First, we describe $GF(3^{97})$ as $GF(3)[x]/(x^{97} + x^{16} + 2)$, where the irreducible polynomial $x^{97} + x^{16} + 2 \in GF(3)[x]$ is well used for the fast implementation of field operations. An element in $GF(3^{97})$ is represented by $\sum_{i=0}^{96} d_i x^i$, where $d_i \in GF(3) = \{0, 1, 2\}$. To transform π and e to an element in $GF(3^{97})$ respectively, we define a bijective map $\phi : \sum_{i=0}^{96} d_i x^i \mapsto \sum_{i=0}^{96} d_i 3^i \in \mathbb{Z}$. We then transform π and e to the 3-adic integer of 97 digits by $\lfloor \pi \cdot 3^{95} \rfloor$ and $\lfloor e \cdot 3^{96} \rfloor$, respectively.

From these values, we define $\mathcal{Q}_\pi = (x_\pi, y_\pi)$ and $\mathcal{Q}_e = (x_e, y_e) \in G_1$ as follows. We first find the non-negative smallest 3-adic integers c_π and c_e such that $\phi^{-1}(\lfloor \pi \cdot 3^{95} \rfloor + c_\pi)$ and $\phi^{-1}(\lfloor e \cdot 3^{96} \rfloor + c_e)$ become x-coordinates of the elements $\mathcal{Q}_\pi$ and $\mathcal{Q}_e$ in the subgroup G_1 on the E. In fact we can set $x_\pi = \phi^{-1}(\lfloor \pi \cdot 3^{95} \rfloor + (11)_3)$ and $x_e = \phi^{-1}(\lfloor e \cdot 3^{96} \rfloor + (120)_3)$. There are two points in $G_1 \backslash \{\mathcal{O}\}$ of the same x-coordinate. We then set the corresponding y-coordinate by computing $y_\pi = (x_\pi^3 - x_\pi + 1)^{(3^{97}+1)/4}$ and $y_e = (x_e^3 - x_e + 1)^{(3^{97}+1)/4}$ in $GF(3^{97})$, respectively.

Again, our goal is to solve the ECDLP in G_1, i.e., for given $Q_\pi, Q_e \in G_1$ we try to find integer s such that $Q_\pi = [s]Q_e$. On the other hand, the η_T pairing enables us to reduce the ECDLP in G_1 to the DLP over G_2 by the relationship $\eta_T(\mathcal{Q}_\pi, \mathcal{Q}_\pi) = \eta_T(\mathcal{Q}_\pi, \mathcal{Q}_e)^s$. Therefore, we can find s by computing the discrete logarithm

$$s = \log_{\eta_T(\mathcal{Q}_\pi, \mathcal{Q}_e)} \eta_T(\mathcal{Q}_\pi, \mathcal{Q}_\pi) = \log_g \eta_T(\mathcal{Q}_\pi, \mathcal{Q}_\pi) / \log_g \eta_T(\mathcal{Q}_\pi, \mathcal{Q}_e) \bmod P_{151},$$

for a generator g of G_2.

3.2 Parameter Settings for FFS

In this section, we explain the parameter setting used for our implementations of the FFS. Hayashi *et al.* [20] reported that, when $n \leq 509$, the JL06-FFS [24] is more efficient for solving the DLP over $GF(3^{6n})$ than the JL02-FFS [23]. Thus, we use the JL06-FFS for our computation. In the JL06-FFS, the condition that "r is monic" is introduced into the collecting relations phase in order to compute efficiently. For the remainder of this paper, the FFS refers to the JL06-FFS.

To solve our DLP over $GF(3^{6\cdot 97})$, we have to select several parameter values of the FFS, such that its computational time is small enough for a fixed extension degree n. The parameter values for $n = 97$ are listed in [31, Table 3], and we use those parameter values for our computation.

We select the parameter $\kappa \in \{1, 2, 3, 6\}$ as follows. $GF(3^{6\cdot 97})$ is described as $GF(3^{\kappa})[x]/(f)$, where $f \in GF(3^{\kappa})[x]$ is an irreducible polynomial of degree $6\cdot 97/\kappa$. The appropriate value of κ is given in [31, Table 3], i.e., $\kappa = 6$. However, we select $\kappa = 3$ for the following reasons. In the linear algebra phase, filtering [15] is performed to reduce the size of the matrix. Then it is required that all elements in the factor base correspond to the memory addresses of the PC for efficient computation. The number of elements in the factor base for $\kappa = 6$ is much larger than that for $\kappa = 3$, so $\kappa = 3$ is advantageous on this point. Additionally, [31, Table3] shows that the computational cost of the FFS for $\kappa = 3$ is only about twice as much as that for $\kappa = 6$. We conducted test runs for $\kappa = 3, 6$ in the collecting relations phase, then noticed that our implementation for $\kappa = 3$ was much faster than for $\kappa = 6$, so we set $\kappa = 3$.

Polynomial Selection Phase: We select the bivariate polynomial $H(x, y)$ of the form $x + y^{d_H}$ for a given parameter d_H of the FFS in the same manner as [20]. Then we search an irreducible polynomial $f \in GF(3^{\kappa})[x]$ and a polynomial $m \in GF(3^{\kappa})[x]$ which are satisfying the condition (1), by factoring $H(x, m)$ for a randomly picked polynomial m whose degree is d_m. In fact, we randomly pick up m from $GF(3)[x]$, so that f is also in $GF(3)[x]$ for use of the Galois action. From [31, Table 3], we set d_H and d_m as 6 and 33, respectively.

Next, we select the smoothness bound $B = 6$ by using [31, Table 3] for (2) and (3), i.e., a rational factor base $F_R(B)$ and an algebraic factor base $F_A(B)$. $\#F_R(B)$ is 67576068 and $\#F_A(B)$ is 67572597, thus the number of elements of factor base, i.e., $\#F_R(B) + \#F_A(B)$, is 135148665.

Collecting Relations Phase: In the collecting relations phase, we use the lattice sieve [29] and the free relation [20] and collect many relations (8); $(r, s) \in (GF(3^{\kappa})[x])^2$ satisfying (4), (5), (7), where r is monic. The search range for the lattice sieve depends on the maximum degrees R, S of r, s. We set $R = S = 6$ based on [31, Table 3]. The lattice sieve gives a certain amount of relations for one special-Q, which is defined in Section 4.1. Therefore, we require a sufficient number of special-Q's so that the number of relations obtained in the collecting relations phase is larger than that of all elements in the factor base. The minimum sufficient number of special-Q's is estimated by the following process. We have to select special-Q's from the subset $F_R(6)\backslash F_R(5)$, whose cardinality is 64566684. Let θ_{min} be the minimum sufficient ratio of special-Q's over all elements in $F_R(6)\backslash F_R(5)$. For $n = 97$ and $\kappa = 3$, we can estimate $\theta_{min} = 0.01292$ [31, Table 3]. Therefore, the number of special-Q's must be larger than $\lceil 0.01292 \cdot 64566684 \rceil = 834202$. In our computation, we set 2500000 as the number of special-Q's to obtain more relations than we require since we expect that these excess relations will help us reduce the size of the matrix during filtering, especially in singleton-clique.

4 Implementation

In this section, we propose the following efficient implementation techniques; the lattice sieve for the JL06-FFS and optimization for our parameters in the collecting relations phase, the data structure and the parallel Lanczos method for the Galois action in the linear algebra phase, for reducing the computational cost of the FFS for solving the DLP over $GF(3^{6 \cdot 97})$. Parameters $(\kappa, d_H, d_m, B, R, S)$ are fixed as $(3, 6, 33, 6, 6, 6)$. The reasoning for this is explained in Section 3.2.

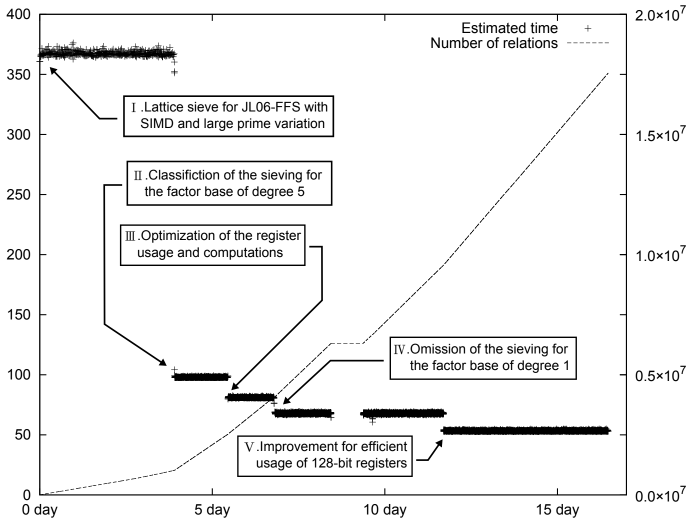
4.1 Collecting Relations Phase

In the collecting relations phase, we used the lattice sieve [29] in a similar fashion to factoring a large integer [26] and solving discrete logarithm problems [22, 23]. We give an overview of our implementation of the lattice sieve in the following paragraphs. More details are described in [19].

Lattice Sieve for JL06-FFS: Sieving with the lattice sieve is performed for $(r, s) \in (GF(3^3)[x])^2$ such that the formula (5) given in Section 2.2 is divisible by an element Q chosen from a subset of the rational factor base $F_R(6) \backslash F_R(5)$ (this Q is called a "special-Q"). Recall that $\deg r$ and $\deg s$ are not greater than $R = 6$ and $S = 6$, respectively. Such (r, s) can be represented as $(r, s) = c(r_1, s_1) + d(r_2, s_2)$ for given reduced lattice bases $(r_1, s_1), (r_2, s_2) \in (GF(3^3)[x])^2$ and any $c, d \in GF(3^3)[x]$ such that $\deg(cr_1 + dr_2) \leq 6, \deg(cs_1 + ds_2) \leq 6$, then sieving is done on the bounded c-d plane. After sieving, we conduct the smoothness test [16] for "candidates" that are evaluated as B-smooth pairs with high probability by using the lattice sieve.

A problem of applying the lattice sieve to the FFS is the condition "r is monic" described in Section 3.2. Since r is represented as $cr_1 + dr_2$, it is difficult to efficiently keep r monic — it might require degree evaluations and branches. Instead of choosing monic r, we introduce the condition $r \equiv 1 \bmod x$. To satisfy this condition, we restrict r_1 and r_2 such that $r_1 \equiv 0 \bmod x$ and $r_2 \equiv 1 \bmod x$. Then sieving is performed on the bounded c-d plane with restriction $d \equiv 1 \bmod x$, whose size is reduced to $1/27$ compared with the original bounded c-d plane. This sieving procedure with the restricted condition can be implemented without extra costs such as additional degree evaluations and additional branches.

Lattice Sieve with SIMD: Since operations of $GF(3)$ can be represented using logical instructions [25], operations of $GF(3^3)[x]$ can be performed using a combination of logical and shift instructions. This means SIMD implementation is appropriate for efficient computation of the lattice sieve. We represent $GF(3^3)$ as polynomial basis $GF(3)[\omega]/(\omega^3 - \omega - 1)$, and its element is represented using



vertical(left) : estimation days for collecting relations phase
vertical(right) : number of collected relations
horizontal : first two weeks of computing days for collecting relations phase
(Period with no data between 8-9 days was due to human error in operating PC.)

Fig. 1. Our improvement in collecting relations phase for first two weeks

6-bit $(h_1, \ell_1, h_\omega, \ell_\omega, h_{\omega^2}, \ell_{\omega^2}) \in GF(2)^6$ in our implementation. We then pack 16 elements of $GF(3^3)[x]$ of degree at most 7 into 6 registers of 128 bits, and treat 16 elements with SIMD. Note that the upper bound of the degree of our SIMD data structure is for efficient access to each element in $GF(3^3)[x]$. On the other hand, since we choose B, R, S as all 6, the upper bound of the degrees of $c, d, r_1, s_1, r_2, s_2 \in GF(3^3)[x]$ and $\mathfrak{p}$ in the factor base, which are treated in the lattice sieve, is also 6. Therefore, our SIMD structure can be stored elements treated in the lattice sieve.

History of Our Optimizations: Figure 1 shows the process of our improvements in the collecting relations phase for the first two weeks. We improved our implementation of the lattice sieve four times during this period. We first used large prime variation to omit sieving for the factor base of degree 6 and implemented the lattice sieve for the FFS with SIMD implementation. We then ran the program for the first four days (stage I in Fig. 1). At that point, the estimated total number of days for the collecting relations phase was about 360 days. While the sieving program was running, we found that sieving for the factor base of degree 5 requires heavier computation than sieving for the factor bases of degree 1, 2, 3 and 4. Therefore, we improved sieving for the factor base of degree 5; thus, our sieving program became over 3 times faster than before (stage II in Fig. 1). Next, we optimized register usage for input values and omitted wasteful computations (stage III in Fig. 1). Additionally, we omitted sieving for the factor base of degree 1 (stage IV in Fig. 1), since that computational time was larger than that for the factor bases of degree 2, 3, 4, and 5. Moreover, we improved our sieving program to use 128-bit registers more efficiently (stage V in Fig. 1). Finally, our sieving program became about 6 times faster than the first one (stage I in Fig. 1) and the estimated total number of days for the collecting relations phase became about 53.1 days. In the next paragraph, we explain the details of the improvement in stage II, which is the most effective and important improvement in our implementation of the lattice sieve.

Details of Stage II: In the lattice sieve, the main computation of sieving for given lattice bases (r_1, s_1), $(r_2, s_2) \in (GF(3^3)[x])^2$ is as follows. For fixed $d \in GF(3^3)[x]$, whose degree is upper-bounded by a degree bound D, we compute $c_0 \equiv -d(r_1t+s_1)^{-1}(r_2t+s_2) \bmod \mathfrak{p}$ for all pairs $(\mathfrak{p}, t) \in \{(\mathfrak{p}, t) \mid \mathfrak{p} \in F_R(B), t \equiv m \pmod{\mathfrak{p}}\} \cup \{(\mathfrak{p}, t) \mid \langle \mathfrak{p}, y - t\rangle \in F_A(B)\}$, and compute $c \in GF(3^3)[x]$, whose degree is upper-bounded by a degree bound C, such that $c = c_0 + k\mathfrak{p}$ where $k \in GF(3^3)[x]$. We call the computation "sieving at d" in this section. For given lattice bases, sieving at d is performed for all d of degree not larger than D. Note that c_0 does not need to be computed when $(r_1t + s_1) \equiv 0 \pmod{\mathfrak{p}}$; therefore we assume $(r_1t + s_1) \not\equiv 0 \pmod{\mathfrak{p}}$ in the following description.

In stage I of our implementation, we found that the time of sieving at d for $\deg \mathfrak{p} = 5$ takes over 100 msec, but each sieving time at d for $\deg \mathfrak{p} = 1, 2, 3$ and 4 takes about 10 mesc or less. Therefore, we tried to improve the sieving of degree 5. When we compute c_0 for $\mathfrak{p}$ of degree 5, the degree of c_0 becomes 4 with probability about 26/27. On the other hand, the degree of the lattice bases r_1, s_1, r_2, s_2 is 3 in most cases because the degree of special-Q is 6. On such bases, degree bounds C and D can be chosen as 3 to satisfy condition (4), i.e., $\deg r \leq 6$ and $\deg s \leq 6$. These facts show that about 26/27 of the computation of sieving for $\mathfrak{p}$ of degree 5 are waste computations. Therefore, we discuss how to sieve only with the polynomial c_0, whose degree is not larger than 3, as follows.

Let $\alpha \in GF(3^3)[x]$ be $-(r_1t+s_1)^{-1}(r_2t+s_2) \bmod \mathfrak{p}$, then we have $c_0 = d\alpha \bmod \mathfrak{p}$. Let $\alpha_i \in GF(3^3)$ be the coefficient of the fourth-order term of $x^i\alpha \bmod \mathfrak{p}$ for $i = 0, 1, 2, 3$. Since $\deg d \leq 3$, d is represented as $d_3x^3 + d_2x^2 + d_1x + 1$ for $d_3, d_2, d_1 \in GF(3^3)$. Recall that we restricted $d \equiv 1 \bmod x$ in our implementation of the lattice sieve. Here we know that the degree of c_0 is not larger than 3 if $d_3\alpha_3 + d_2\alpha_2 + d_1\alpha_1 + \alpha_0 = 0$. Therefore, it is sufficient to perform sieving at d for $\mathfrak{p}$ in the factor base of degree 5 for only d satisfying the following property:

$$d_1 = \begin{cases} -K\alpha_1^{-1} & \text{if } \alpha_1 \neq 0 \\ \text{any element in } GF(3^3) & \text{if } \alpha_1 = 0 \text{ and } K = 0 \end{cases} \tag{9}$$

where $K = d_3\alpha_3 + d_2\alpha_2 + \alpha_0$. When $\alpha_1 = 0$ and $K = 0$, we should compute c_0 for d whose d_1 is any element in $GF(3^3)$, and we cannot cut off any d_1; therefore, we assume that $\alpha_1 \neq 0$ in the following description. Suppose that we now fix lattice bases $(r_1, s_1), (r_2, s_2)$ and a pair $(\mathfrak{p}, t)$ where $\deg \mathfrak{p} = 5$, then each α_i for $i = 0, 1, 2, 3$ is also fixed. Therefore, since K depends on d_2 and d_3, the d_1 satisfying (9) is given by d_2 and d_3 and uniquely determined for given

d_2 and d_3. This implies that, since d_1 is in $GF(3^3)$ whose cardinality is 27, we can ignore 26 d_1's not satisfying (9) for given d_2 and d_3. In fact, the time of sieving at d for all pairs $(\mathfrak{p}, t)$ where $\deg \mathfrak{p} = 5$ is reduced to about 1.5 msec by ignoring d_1 not satisfying (9). Note that we need to compute K for given d_2 and d_3 for all pairs $(\mathfrak{p}, t)$. The time of computing K for all $(\mathfrak{p}, t)$ takes about 150 msec in our implementation. Therefore, for all pairs $(\mathfrak{p}, t)$ where $\deg \mathfrak{p} = 5$, the computations of K and sieving at d require about 7.1 msec at stage II, which is over 10 times faster than the computation of sieving at d at stage I. As a result, our implementation of the lattice sieve at stage II becomes over 3 times faster than that at stage I.

4.2 Linear Algebra Phase

After the collecting relations phase, we obtain a system of linear equations modulo P_{151}, which is described in Section 2.1. The Galois action [20, 24] can reduce the number of variables of the system of linear equations to one-third. Additionally, after the Galois action, the numbers of equations and variables of the system of linear equations can be further reduced using filtering [15], i.e., singleton-clique and merging. To solve the system of linear equations defined by this reduced matrix, we use the parallel Lanczos method [4, 20].

Galois Action: The Galois action to $GF(3^{6\cdot 97})/GF(3^{3\cdot 97})$ enables us to reduce the number of variables of the system of linear equations to one-third (details of the Galois action are discussed in [20, 24]). However, when we use the Galois action, 151-bit large integers such as $e_0 + e_1\tau + e_2\tau^2$, where $\tau = 3^{97^2} \bmod P_{151}$ and e_i is a small integer of a few bits, are added to elements of the system of linear equations. This unfortunate fact eventually increases the data size of the reduced matrix; therefore, high-capacity memory is required. To allay the increase in the representation size of the elements, we store only a triplet (e_1, e_2, e_3) in the PC memory, not a large 151-bit integer. Since e_i is small enough to be represented by 8 bits, the size of the elements is reduced from 151 to 24 bits on average. We call this representation the "τ-adic structure". Note that the τ-adic structure is used for the Galois action and singleton-clique.

Singleton-Clique and Merging: Filtering consists of two parts, singleton-clique and merging. Singleton-clique deletes unnecessary rows and columns to reduce the size of the matrix. In our implementation of singleton-clique, we performed by maintaining 20000 more rows than columns to prevent accidentally decreasing the rank of the matrix. After that, merging, a weight-controlled Gaussian elimination, is performed. In merging, for small integer k, the column with a weight not larger than k is deleted by row elimination with controlling the pivot selection so that the weight of the matrix is as small as possible. This operation is called k-way merging. In our implementation of merging, we converted the data representation of the matrix from the τ-adic structure to a large 151-bit integer structure, since merging on the τ-adic structure cannot reduce the size of the matrix enough due to the restriction of the pivot selection. More details are described in [19].

Parallel Lanczos Method: By using the parallel Lanczos method [4, 20], we solve the system of linear equations defined by the matrix reduced via the Galois action, singleton-clique, and merging. For parallel computing, the matrix should be split into sub-matrices, i.e., split into $N = N_1 \times N_2$ sub-matrices for N nodes, and nodes communicate among N_1 nodes or N_2 nodes. To reduce the synchronization time before communicating, the matrix is split so that each sub-matrix has almost the same weight. Our machine environment for the parallel Lanczos method consisted of 22 nodes, and each node had 12 CPU cores and 2 NICs. The 2 NICs were connected to a 48-port Gbit HUB, i.e., 44 ports were used for connecting 22 nodes. All 22 nodes could be used, so we had a choice for machine environment; $20 = 5 \times 4$, $21 = 7 \times 3$ or $22 = 11 \times 2$. Using 20 nodes requires the least communication costs but the most computational costs, and using 22 nodes requires the most communication costs but the least computational costs. Using 21 nodes was the best for our implementation; therefore, we used 21 nodes.

For computation in the parallel Lanczos method, many modular multiplications of 151-bit integers $\times$ 151-bit integers modulo P_{151} are required due to the Galois action. We implemented Montgomery multiplication optimized to 151-bit integers using assembly language. Our program then becomes several times faster than straightforward modular multiplication using GMP (`http://gmplib.org/`) for multiple precision arithmetic.

After the computation of the parallel Lanczos method started, we improved our codes of the parallel Lanczos method (for example, efficient register usage, overlapping communications and computations). These improvements are about 15% faster than our initial implementation.

4.3 Individual Logarithm Phase

As mentioned in Section 3.1, $\log_g \eta_T(\mathcal{Q}_\pi, \mathcal{Q}_\pi)$ and $\log_g \eta_T(\mathcal{Q}_\pi, \mathcal{Q}_e)$ are required to solve our target problem. To compute them, rationalization and special-Q descent [24] were used. For simplicity, let T be $\eta_T(\mathcal{Q}_\pi, \mathcal{Q}_\pi)$, or $\eta_T(\mathcal{Q}_\pi, \mathcal{Q}_e)$ in the following paragraphs.

In the rationalization, we randomize T such that the randomized element is M-smooth for a small enough integer $M > B$ by the following process. First, we randomize T by $z \equiv g^\gamma T \pmod{f}$ for a random integer $\gamma \in \mathbb{Z}_{P_{151}}$. We then rationalize z as $z \equiv z_1/z_2 \pmod{f}$ where degrees of z_1 and z_2 are about $\deg f/2$, and check whether both z_1 and z_2 are M-smooth. Then, computing $\log_g T$ is reduced to computing logarithms of irreducible factors of M-smooth elements z_1 and z_2.

M-smooth elements z_i for $i = 1, 2$, contain some irreducible factors of degree larger than B whose logarithms are not computed in the linear algebra phase. To compute these logarithms, the special-Q descent [24] is usually used. In the special-Q descent, the lattice sieve is recursively conducted with an irreducible factor of degree larger than B, which is contained in z_i or in a relation generated during the special-Q descent, as a special-Q.

5 Experimental Results

We succeeded in solving a DLP over $GF(3^{6\cdot 97})$ by using the FFS with our efficient implementation techniques discussed in Section 4. In this section, we report our computational results, such as the computational time of each phase of the FFS and the number of relations.

5.1 Polynomial Selection

The FFS has six parameters κ, d_H, d_m, B, R, and S, as defined in Section 2.2, and we set $(\kappa, d_H, d_m, B, R, S) = (3, 6, 33, 6, 6, 6)$ for our target problem, based on the reason given in Section 3.2. In the polynomial selection phase, we can extract appropriate polynomials such as the definition polynomial $H(x, y)$ of a function field described in Section 3.2 in one minute, so the computational cost of the polynomial selection phase is negligibly small.

5.2 Collecting Relations Phase

In the collecting relations phase, we search many relations that are equations of the form (8) to generate a system of linear equations by using the lattice sieve and the free relation. We explain our computational results of the collecting relations phase, e.g., the number of relations obtained in this phase, the computational time of the lattice sieve for one special-Q.

Lattice Sieve. Each special-Q has to be chosen from $F_R(6)\backslash F_R(5)$. The number of elements of $F_R(6)\backslash F_R(5)$ is 64566684, and the size of the table of those elements is about 500 MB. Since our program of the lattice sieve is computed using many nodes, it is not convenient to pick up the element from that 500-MB table as a special-Q. Therefore, we selected a special-Q by randomly generating an irreducible polynomial in $GF(3^3)[x]$ of degree 6, which is in $F_R(6)\backslash F_R(5)$, and iterated the computation of the lattice sieve for the special-Q.

We prepared 47 PCs (in total 212 CPU cores) for the lattice sieve. The computation of the lattice sieve began on May 14, 2011, and we continued optimizing our program of the collecting relations phase. As discussed in Section 4.1, we applied several improvements to our program of the collecting relations phase; the lattice sieve for the JL06-FFS, the lattice sieve with SIMD, and optimization for our parameters. Figure 1 in Section 4.1 shows the process of our improvements in the collecting relations phase for the first two weeks. The total time for the collecting relations phase shortened due to our improvements. Finally, the computation finished on September 9, 2011 and required 118 days. including the loss-time of some programming errors, updating our codes, and power outages. The real computational time of the lattice sieve was equivalent to 53.1 days using 212 CPU cores such as Xeon E5440.

Table 2 summarizes the process of generating relations in the collecting relations phase. It might seem that the number of duplicate relations is very small compared to the integer factorization case using the number field sieve. This

Table 2. Number of collected relations in collecting relations phase

lattice sieve	159032292 relations obtained from 2500000 special-Q's (64.91 relations/special-Q, 389 sec/special-Q)
	153815493 unique (non-duplicated) relations obtained from 2449991 unique special-Q's
free relation	33786299 relations
total	187602242 relations (consist of 134697663 elements in the factor base)

Table 3. Compressing matrix using Galois action, singleton-clique and merging

method	size of matrix
before compressing	187602242 equations $\times$ 134697663 variables
Galois action	159394665 equations $\times$ 45049572 variables
singleton-clique	14060794 equations $\times$ 14040791 variables
6-way merging	6141443 equations $\times$ 6121440 variables

arises from the fact that the size of the sieving space in our parameters is so large compared to that case.

Free Relation. The free relation gives us additional relations not generated by a sieving algorithm such as the lattice sieve. The details of the free relation is given in [20]. As shown in Table 2, the free relation gave us 33786299 relations. Eventually, we obtained a system of linear equations consisting of 187602242 equations and 134697663 variables. Note that there are 451002 elements in the factor base, which does not appear in the 187602242 relations.

5.3 Linear Algebra Phase

In the linear algebra phase, we firstly reduced the size of the matrix by the Galois action and filtering, and then performed the parallel Lanczos method for the reduced matrix. Table 3 shows that the process of the compression of the matrix.

Galois Action. As mentioned in Section 4.2, the Galois action reduced the size of the matrix generated in the collecting relations phase to one-third since $\kappa = 3$. To allay the fact that the size of each element of the matrix increases from a few bits to 151 bits due to the Galois action, we used the τ-adic structure mentioned in Section 4.2.

Singleton-Clique and Merging. After using the Galois action, we additionally reduce the variables and equations of the matrix by singleton-clique and merging [15]. Using a PC, the computation for singleton-clique took about 3 hours, and that for merging took about 10 hours. After 6-way merging, we started the computation of the parallel Lanczos method for the 6-way merged matrix. See [19] for more details about our results of singleton-clique and merging.

Table 4. Computational time of parallel Lanczos method for 6-way merged matrix

calculation time/loop	626.3 msec
synchronization time/loop	46.5 msec
communication time/loop	457.3 msec
total time/loop	1130.1 msec
number of loops	6121438
total time	80.1 days

Parallel Lanczos Method. We used the parallel Lanczos method [4, 20] to solve the system of linear equations defined by the 6-way merged matrix. Note that this matrix is sparse and defined over $\mathbb{Z}_{P_{151}}$, where P_{151} is the 151-bit prime number given in Section 3.1. The computation of the parallel Lanczos method started on January 16, 2012, and was conducted on 21 PCs, which were connected via a 48-port Gbit HUB. As mentioned in Section 4.2, we continued improving our codes of the parallel Lanczos method after computation began. The computational times of our improved codes are listed in Table 4. Finally, computation finished on April 14, 2012. The computation for the parallel Lanczos method took 90 days including time losses similar to our implementation of the lattice sieve. The real computational time is equivalent to 80.1 days using 252 CPU cores such as Xeon X5650.

5.4 Individual Logarithm Phase

Our target is to compute $\log_g \eta_T(\mathcal{Q}_\pi, \mathcal{Q}_e)$ and $\log_g \eta_T(\mathcal{Q}_\pi, \mathcal{Q}_\pi)$ for some $g \in G_2$, as mentioned in Section 3.1.

First, we computed the rationalization described in Section 4.3. Let g be a polynomial $(x+\omega)^{(3^{6\cdot 97}-1)/P_{151}} \in G_2$, where ω is a polynomial basis of $GF(3^3) \cong GF(3)[\omega]/(\omega^3-\omega-1)$. Note that g is a generator of $G_2 \subset GF(3^{6\cdot 97})^*$ and $x+\omega$ is a monic irreducible polynomial in $F_R(B)$ of degree 1. We set $M = 15$ and search a pair (z_1, z_2) (and (z_1', z_2')) $\in (GF(3^3)[x])^2$ such that $\eta_T(\mathcal{Q}_\pi, \mathcal{Q}_e)\cdot g^{\gamma_1} = z_1/z_2$ (and $\eta_T(\mathcal{Q}_\pi, \mathcal{Q}_\pi) \cdot g^{\gamma_2} = z_1'/z_2'$), where z_i (and z_i') are M_i-smooth (where $M_i \leq M$) for some $\gamma_1, \gamma_2 \in \mathbb{Z}_{P_{151}}$ and $i = 1, 2$. We found z_1 and z_2, which are 13- and 15-smooth (and z_1' and z_2' which are 15- and 14-smooth), respectively. These computations were conducted on 168 CPU cores and required 7 days for each computation.

$$\begin{aligned}
\eta_T(\mathcal{Q}_\pi, \mathcal{Q}_e) \cdot g^{\gamma_1} &= (13\text{-smooth})/(15\text{-smooth}),\\
\gamma_1 &= 2514037766787322013334785428291787565870435706,\\
\eta_T(\mathcal{Q}_\pi, \mathcal{Q}_\pi) \cdot g^{\gamma_2} &= (15\text{-smooth})/(14\text{-smooth}),\\
\gamma_2 &= 2657516740789758289434702436228062607247517136.
\end{aligned}$$

Next, we performed special-Q descent for each irreducible factor of smooth elements obtained by the rationalization. These computations were conducted on 168 CPU cores and took about 0.5 days for each $\eta_T(\mathcal{Q}_\pi, \mathcal{Q}_e)$ and $\eta_T(\mathcal{Q}_\pi, \mathcal{Q}_\pi)$.

Thus, the computation of the individual logarithm phase took 15 days; (7 days (for rationalization) + 0.5 days (for special-Q descent)) $\times$ 2 elements.

By using the logarithms of the corresponding elements in the factor base obtained from the linear algebra phase, we could compute $\log_g \eta_T(\mathcal{Q}_\pi, \mathcal{Q}_e)$ and $\log_g \eta_T(\mathcal{Q}_\pi, \mathcal{Q}_\pi)$. The logarithm of each element is as follows:

$$\begin{aligned}\log_g \eta_T(\mathcal{Q}_\pi, \mathcal{Q}_e) &= 1540966625957007958347823268423957036469656370,\\ \log_g \eta_T(\mathcal{Q}_\pi, \mathcal{Q}_\pi) &= 1630281950635507295663809171217833096970449894.\end{aligned}$$

Finally, we obtained the logarithm of the target element:

$$\begin{aligned}s &= \log_{\eta_T(\mathcal{Q}_\pi,\mathcal{Q}_e)} \eta_T(\mathcal{Q}_\pi, \mathcal{Q}_\pi)\\ &= 1752799584850668137730207306198131424550967300.\end{aligned}$$

This is the solution of the ECDLP of equation $\mathcal{Q}_\pi = [s]\mathcal{Q}_e$.

6 Concluding Remarks

We evaluated the security of pairing-based cryptosystems using the η_T pairing over supersingular elliptic curves on finite field $GF(3^n)$. We focused on the case of $n = 97$ since many implementers have reported practically relevant high-speed implementations of the η_T pairing with $n = 97$ in both software and hardware. In particular, we examined the difficulty in solving the discrete logarithm problem (DLP) over $GF(3^{6\cdot 97})$ by our implementation of the function field sieve (FFS).

To reduce the computational cost of the FFS for solving the DLP, we proposed several efficient implementation techniques. In the collecting relations phase, we implemented the lattice sieve for the JL06-FFS with SIMD and introduced improvements by optimizing for factor bases of each degree; therefore, our lattice sieve for the JL06-FFS became about 6 times faster than the first one. The main difference from the number field sieves for integer factorization is the linear algebra phase, namely, we have to deal with a large modulus of 151-bit prime for the computation of the FFS. We thus performed filtering (singleton-clique and merging) by carefully considering the data structure of large integers developing from the Galois action, so that we can efficiently conduct the parallel Lanczos method. From the above improvements, we succeeded in solving the DLP over $GF(3^{6\cdot 97})$ in 148.2 days by using PCs with 252 CPU cores. Our computational results contribute to the security estimation of pairing-based cryptosystems using the η_T pairing. In particular, they show that the security parameter of such pairing-based cryptosystems must be chosen with $n > 97$.

Finally, we show a very rough estimation of required computational power for solving the DLP over $GF(3^{6n})$ with $n > 97$. Our experiment on the DLP over $GF(3^{6n})$ with $n = 97$ used 252 CPU cores of mainly 2.67 GHz Xeon for 148.2 days, which are equivalent to $2^{62.9}$ clock cycles. From the analysis of [31], the computational complexities of breaking the DLP over $GF(3^{6n})$ with $n = 163$ and 193 become $2^{15.4}$ and $2^{19.1}$ times larger than that with $n = 97$, respectively. Therefore, we could estimate that about $2^{78.3}$ and $2^{82.0}$ clock cycles are required for breaking the DLP over $GF(3^{6n})$ with $n = 163$ and 193, respectively. On the other hand, the currently second fastest supercomputer K has a throughput of about 10.5 petaflop/s from `http://www.top500.org/`, and it performs about $2^{78.1}$ floating-point operations for one year. If one floating-point operation on the CPU of the K is equivalent to one clock cycle of logical operation on the Xeon core, we might be able to break the DLP over $GF(3^{6\cdot 163})$ using our implementation on supercomputer K for one year.

References

1. Adleman, L.M.: The Function Field Sieve. In: Huang, M.-D.A., Adleman, L.M. (eds.) ANTS 1994. LNCS, vol. 877, pp. 108–121. Springer, Heidelberg (1994)
2. Adleman, L.M., Huang, M.-D.A.: Function field sieve method for discrete logarithms over finite fields. Inform. and Comput. 151, 5–16 (1999)
3. Ahmadi, O., Hankerson, D., Menezes, A.: Software Implementation of Arithmetic in F_{3^m}. In: Carlet, C., Sunar, B. (eds.) WAIFI 2007. LNCS, vol. 4547, pp. 85–102. Springer, Heidelberg (2007)
4. Aoki, K., Shimoyama, T., Ueda, H.: Experiments on the Linear Algebra Step in the Number Field Sieve. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) IWSEC 2007. LNCS, vol. 4752, pp. 58–73. Springer, Heidelberg (2007)
5. Barreto, P.S.L.M., Galbraith, S., ÓhÉigeartaigh, C., Scott, M.: Efficient pairing computation on supersingular Abelian varieties. Des., Codes Cryptogr. 42(3), 239–271 (2007)
6. Barreto, P.S.L.M., Kim, H.Y., Lynn, B., Scott, M.: Efficient Algorithms for Pairing-Based Cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–368. Springer, Heidelberg (2002)
7. Beuchat, J.-L., Brisebarre, N., Detrey, J., Okamoto, E.: Arithmetic Operators for Pairing-Based Cryptography. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 239–255. Springer, Heidelberg (2007)
8. Beuchat, J.-L., Brisebarre, N., Detrey, J., Okamoto, E., Shirase, M., Takagi, T.: Algorithms and arithmetic operators for computing the η_T pairing in characteristic three. IEEE Trans. Comput. 57(11), 1454–1468 (2008)
9. Beuchat, J.-L., Brisebarre, N., Shirase, M., Takagi, T., Okamoto, E.: A Coprocessor for the Final Exponentiation of the η_T Pairing in Characteristic Three. In: Carlet, C., Sunar, B. (eds.) WAIFI 2007. LNCS, vol. 4547, pp. 25–39. Springer, Heidelberg (2007)
10. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
11. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
12. Boneh, D., Gentry, C., Waters, B.: Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)
13. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)

14. Bos, J.W., Kaihara, M.E., Kleinjung, T., Lenstra, A.K., Montgomery, P.L.: Solving a 112-bit prime elliptic curve discrete logarithm problem on game consoles using sloppy reduction. International Journal of Applied Cryptography 2(3), 212–228 (2012)
15. Cavallar, S.: Strategies in Filtering in the Number Field Sieve. In: Bosma, W. (ed.) ANTS-IV. LNCS, vol. 1838, pp. 209–231. Springer, Heidelberg (2000)
16. Gordon, D.M., McCurley, K.S.: Massively Parallel Computation of Discrete Logarithms. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 312–323. Springer, Heidelberg (1993)
17. Granger, R., Page, D., Stam, M.: Hardware and software normal basis arithmetic for pairing-based cryptography in characteristic three. IEEE Trans. Comput. 54(7), 852–860 (2005)
18. Hankerson, D., Menezes, A., Scott, M.: Software implementation of pairings. In: Identity-Based Cryptography, pp. 188–206 (2009)
19. Hayashi, T., Shimoyama, T., Shinohara, N., Takagi, T.: Breaking pairing-based cryptosystems using η_T pairing over $GF(3^{97})$. Cryptology ePrint Archive, Report 2012/345 (2012)
20. Hayashi, T., Shinohara, N., Wang, L., Matsuo, S., Shirase, M., Takagi, T.: Solving a 676-bit Discrete Logarithm Problem in $GF(3^{6n})$. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 351–367. Springer, Heidelberg (2010)
21. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. In: Bosma, W. (ed.) ANTS-IV. LNCS, vol. 1838, pp. 385–394. Springer, Heidelberg (2000)
22. Joux, A., et al.: Discrete logarithms in $\mathrm{GF}(2^{607})$ and $\mathrm{GF}(2^{613})$. Posting to the Number Theory List (2005), `http://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind0509&L=nmbrthry&T=0&P=3690`
23. Joux, A., Lercier, R.: The Function Field Sieve Is Quite Special. In: Fieker, C., Kohel, D.R. (eds.) ANTS 2002. LNCS, vol. 2369, pp. 431–445. Springer, Heidelberg (2002)
24. Joux, A., Lercier, R.: The Function Field Sieve in the Medium Prime Case. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 254–270. Springer, Heidelberg (2006)
25. Kawahara, Y., Aoki, K., Takagi, T.: Faster Implementation of η_T Pairing over $GF(3^m)$ Using Minimum Number of Logical Instructions for $GF(3)$-addition. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 282–296. Springer, Heidelberg (2008)
26. Kleinjung, T., Aoki, K., Franke, J., Lenstra, A.K., Thomé, E., Bos, J.W., Gaudry, P., Kruppa, A., Montgomery, P.L., Osvik, D.A., te Riele, H., Timofeev, A., Zimmermann, P.: Factorization of a 768-Bit RSA Modulus. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 333–350. Springer, Heidelberg (2010)
27. Menezes, A., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Trans. IT 39(5), 1639–1646 (1993)
28. Okamoto, T., Takashima, K.: Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
29. Pollard, J.M.: The lattice sieve. In: The development of the number field sieve. LNIM, vol. 1554, pp. 43–49 (1993)
30. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
31. Shinohara, N., Shimoyama, T., Hayashi, T., Takagi, T.: Key Length Estimation of Pairing-Based Cryptosystems using η_T Pairing. In: Ryan, M.D., Smyth, B., Wang, G. (eds.) ISPEC 2012. LNCS, vol. 7232, pp. 228–244. Springer, Heidelberg (2012)