Solving a 676-bit Discrete Logarithm Problem in GF(3^{6n})

Takuya Hayashi[1⋆], Naoyuki Shinohara[2], Lihua Wang[2], Shin'ichiro Matsuo[2],
Masaaki Shirase[3], and Tsuyoshi Takagi[1⋆]

[1] Graduate School of Mathematics, Kyushu University,
744, Motooka, Nishi-ku, Fukuoka, 819-0395, Japan.
[2] Information Security Research Center, National Institute of Information and Communications Technology,
4-2-1, Nukui-Kitamachi, Koganei, Tokyo 184-9795, Japan.
[3] School of Systems Information Science, Future University Hakodate,
116-2, Kamedanakano-cho, Hakodate, Hokkaido, 041-0806, Japan.

Abstract. Pairings on elliptic curves over finite fields are crucial for constructing various cryptographic schemes. The η_T pairing on supersingular curves over GF(3^n) is particularly popular since it is efficiently implementable. Taking into account the Menezes-Okamoto-Vanstone (MOV) attack, the discrete logarithm problem (DLP) in GF(3^{6n}) becomes a concern for the security of cryptosystems using η_T pairings in this case. In 2006, Joux and Lercier proposed a new variant of the function field sieve in the medium prime case, named JL06-FFS. We have, however, not yet found any practical implementations on JL06-FFS over GF(3^{6n}). Therefore, we first fulfill such an implementation and we successfully set a new record for solving the DLP in GF(3^{6n}), the DLP in GF($3^{6 \cdot 71}$) of 676-bit size. In addition, we also compare JL06-FFS and an earlier version, named JL02-FFS, with practical experiments. Our results confirm that the former is several times faster than the latter under certain conditions.

Key words: function field sieve, discrete logarithm problem, pairing-based cryptosystems

1 Introduction

Based on pairings, many novel cryptographic protocols have been successively constructed, such as identity-based encryptions [8], forward-secure cryptosystems, proxy cryptosystems, keyword searchable PKEs [7]. As a result, two requirements arose: efficient pairing computation and security parameter selection.

The η_T pairing [5] on supersingular curves over GF(3^n) has been efficiently implemented both in software and hardware [6, 13, 14][1]. Along with the increase

[⋆] This work was done when authors belonged to Future University Hakodate.

[1] Here, n is a prime number such as $n = 97$, 163 and 193 [25].

in computation speed on the η_T pairing, one may ask whether cryptosystems based on the η_T pairing are still secure. It is well known that a discrete logarithm problem (DLP) on supersingular curves over GF(q) can be converted to a DLP in GF(q^m) (where q is a prime power and m is not larger than 6) [24]. Therefore, the DLP in GF(3^{6n}) is one of the most important problems in analyzing the cryptosystems constructed with the η_T pairing on supersingular curves over GF(3^n).

The function field sieve (FFS) is the most efficient algorithm for solving the DLP in finite fields of small characteristic. The complexity of the FFS for solving the DLP in GF(3^{6n}) is $L_{3^{6n}}[1/3,\, c]$ with constant c, where

$$L_{3^{6n}}[1/3,\, c] = \exp((c + o(1))(\log 3^{6n})^{1/3}(\log\log 3^{6n})^{2/3}).$$

Here $o(1)$ stands for a function that converges to zero as n approaches infinity.

The first FFS was proposed by Adleman [1] in 1994. Five years later, Adleman and Huang proposed an improved FFS (AH-FFS) with $c = (32/9)^{1/3}$ [2]. In 2002, Joux and Lercier proposed a practical improvement of the FFS (JL02-FFS) [16]. Since a definition polynomial of the function field in JL02-FFS can select more flexibly, JL02-FFS is more practical than AH-FFS, though its asymptotic complexity is the same as that of AH-FFS. Furthermore, by using JL02-FFS, Joux and Lercier succeeded in solving the DLP in GF(2^{613}). This refreshed the record for solving the DLP in finite fields of characteristic two with regard to bit size [15]. In 2006, Joux and Lercier proposed another new variant of the FFS (JL06-FFS) [18]. JL06-FFS has the same asymptotic complexity with JL02-FFS for solving the DLP in GF(3^{6n}), where n is a prime number[2]. This work implied that JL06-FFS might be efficient for solving the DLP in extension fields of GF(3^6) of degree n. However, to our knowledge, there have been no practical experiments. Note that JL02-FFS can also be applied to extension fields of GF(3^6) of degree n, but [12] showed no advantage using GF(3^6) as the base field.

Our contributions. We have first conducted experiments on JL06-FFS in GF(3^{6n}). In JL06-FFS, GF(3^{6n}) is constructed as extension fields of GF(3^6) of degree n, and thus the Galois action can be dealt for reducing required relations. By our implementation, we succeeded in solving the DLP in GF($3^{6 \cdot 71}$) of 676-bit size with about 33 days computation, which is the new record for solving the DLP in GF(3^{6n}). Our work contributes to the selecting of security parameters. Additionally, we compared JL06-FFS [18] with JL02-FFS [16], and according to the experimental results, we confirmed that JL06-FFS is several times faster than JL02-FFS with $n = 19, 61$.

The rest of the paper is organized as follows. In Section 2, we briefly review the FFS algorithm. In Section 3, we compare JL02-FFS with JL06-FFS according to the polynomial selection method and experimental results. In Section 4, we describe our implementation on how to solve the DLP in GF($3^{6 \cdot 71}$) in detail, which is based on JL06-FFS. Concluding remarks are made in Section 5.

[2] When n is a composite number, this variant may have complexity $L_{3^{6n}}[1/3, 3^{1/3}]$ for solving the DLP in GF(3^{6n}) (When JL06-FFS has complexity $L_{q^m}[1/3, 3^{1/3}]$, we call it JL06-FFS-2). We do not deal with this case in this paper.

2 Outline of Function Field Sieve

In this section, we describe an overview of the FFS [1], which consists of four steps: polynomial selection, collection of relations, linear algebra, and individual logarithm. We particularly deal with the FFS for solving the DLP in extension fields of GF(3^6) of degree n and describe the four steps below. For more details, refer to related work as [1, 12, 16, 18].

Throughout this paper, let γ be a generator of the multiplicative group of GF(3^{6n}) and $\alpha \in \langle \gamma \rangle$, then we try to find the smallest positive integer $\log_\gamma \alpha$ such that $\gamma^{\log_\gamma \alpha} = \alpha$, which is called the discrete logarithm.

1. Polynomial selection: Select $f \in \mathrm{GF}(3^6)[x]$ such that f is a monic irreducible polynomial of degree n, then $\mathrm{GF}(3^{6n}) \cong \mathrm{GF}(3^6)[x]/(f)$. Next, find a polynomial $H(x, y) \in \mathrm{GF}(3^6)[x, y]$ satisfying the eight conditions proposed by Adleman [1]. Then there is a surjective homomorphism

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

where m is in $\mathrm{GF}(3^6)[x]$ such that $H(x, m) \equiv 0 \pmod f$. Here we select the smoothness bound B and define a rational factorbase B_R and an algebraic factorbase B_A as follows:

$$B_R = \{\mathfrak{p} \in \mathrm{GF}(3^6)[x] \mid \deg(\mathfrak{p}) \leq B,\ \mathfrak{p} \text{ is irreducible}\},$$
$$B_A = \{\langle \mathfrak{p}, y - t \rangle \in \mathrm{Div}(\mathrm{GF}(3^6)[x, y]/(H)) \mid \mathfrak{p} \in B_R,\ t \equiv m \pmod{\mathfrak{p}}\},$$

where $\mathrm{Div}(\mathrm{GF}(3^6)[x, y]/(H))$ is the divisor group of $\mathrm{GF}(3^6)[x, y]/(H)$ and $\langle \mathfrak{p}, y - t \rangle$ is a divisor generated by $\mathfrak{p}$ and $y - t$.

2. Collection of relations: For $r, s \in \mathrm{GF}(3^6)[x]$ of degree not larger than B, find at least $(\#B_R + \#B_A)$ relatively prime pairs (r, s) such that

$$\begin{aligned} rm + s &= \prod_{\mathfrak{p}_i \in B_R} \mathfrak{p}_i^{a_i} \\ \langle ry + s \rangle &= \sum_{\langle \mathfrak{p}_j, t_j \rangle \in B_A} b_j \langle \mathfrak{p}_j, y - t_j \rangle. \end{aligned} \tag{1}$$

Such a pair (r, s) is called a double smooth pair. For each (r, s), compute

$$rm + s, \tag{2}$$

$$(-r)^d H(x, -s/r). \tag{3}$$

Polynomial (3) is said to be B-smooth if it is factorized into irreducible polynomials of degree not larger than B, and then we have

$$(-r)^d H(x, -s/r) = \prod_{\langle \mathfrak{p}_j, t_j \rangle \in B_A} \mathfrak{p}_j^{b_j}, \tag{4}$$

where t_j is uniquely determined by r, s and $\mathfrak{p}_j$. Then the b_j in Equation (4) is exactly the same as the one in Equation (1). When both Polynomials (2) and (3) are B-smooth, a pair (r, s) is a double smooth pair. Eventually, we obtain the following relation:

$$\sum_{\mathfrak{p}_i \in B_R} a_i \log_\gamma \mathfrak{p}_i \equiv \sum_{\langle \mathfrak{p}_j, t_j \rangle \in B_A} b_j \log_\gamma \kappa_j \pmod{(3^{6n}-1)/(3^6-1)}, \tag{5}$$

where

$$\kappa_j = \Phi(\lambda_j)^{1/h},\ \langle \lambda_j \rangle = h \langle \mathfrak{p}_j\ y - t_j \rangle, \tag{6}$$

for the class number h of the quotient field of $\mathrm{GF}(3^6)(x)[y]/(H)$.

3. Linear algebra: For the number R of relations, construct an $R \times (\#B_R + \#B_A)$ matrix M from the relations in Equation (5) and $(\#B_R + \#B_A)$ dimensional column vector $\boldsymbol{v}$ as follows:

$$M = \begin{pmatrix} a_1^{(1)} & \dots & a_{\#B_R}^{(1)} & -b_1^{(1)} & \dots & -b_{\#B_A}^{(1)} \\ \vdots & & \vdots & \vdots & & \vdots \\ a_1^{(R)} & \dots & a_{\#B_R}^{(R)} & -b_1^{(R)} & \dots & -b_{\#B_A}^{(R)} \end{pmatrix},\ \boldsymbol{v} = \begin{pmatrix} \log_\gamma \mathfrak{p}_1 \\ \vdots \\ \log_\gamma \mathfrak{p}_{\#B_R} \\ \log_\gamma \kappa_1 \\ \vdots \\ \log_\gamma \kappa_{\#B_A} \end{pmatrix}.$$

Then we solve the linear equation

$$M\boldsymbol{v} \equiv 0 \pmod{(3^{6n}-1)/(3^6-1)}. \tag{7}$$

4. Individual logarithm: Find integers e_i, f_j such that

$$\log_\gamma \alpha \equiv \sum_{\mathfrak{p}_i \in B_R} e_i \log_\gamma \mathfrak{p}_i + \sum_{\langle \mathfrak{p}_j, t_j \rangle \in B_A} f_j \log_\gamma \kappa_j \pmod{(3^{6n}-1)/(3^6-1)},$$

then compute the discrete logarithm $\log_\gamma \alpha$. This is done using the special-$\mathfrak{q}$ descent method [16, 18, 19].

3 Comparison of Polynomial Selection on JL02-FFS and JL06-FFS

The two most efficient variants of the FFS for solving the DLP in $\mathrm{GF}(3^{6n})$ are JL02-FFS and JL06-FFS. Although they have the same asymptotic complexity, there is a considerable difference between them in the fixed extension degree for practical use. The time complexities of JL02-FFS and JL06-FFS depend on the size of each sieving area, which is the number of pairs (r, s), and each size is explained in the following subsections. Note that our comparison is done merely by the size of the sieving area, and the detailed analysis should incorporate the non-integer smoothness bound estimated by Granger [11].

3.1 Polynomial Selection of JL02-FFS and Its Sieving Area

At first we describe an outline of the polynomial selection of JL02-FFS, after that we estimate the size of the sieving area. In order to distinguish from previous section, we set the subindex "02" after the symbols.

Let $H_{02}(x, y)$ of degree d_{02} in y be formed as C_{ab} curves [23]:

$$H_{02}(x, y) = h_{a,0}y^a + h_{0,b}x^b + \sum_{ib+ja<ab} h_{i,j}y^i x^j \quad (h_{i,j} \in \mathrm{GF}(3),\, h_{a,0},\, h_{0,b} \neq 0).$$

Randomly choose polynomials $u_1, u_2 \in \mathrm{GF}(3)[x]$ of degree at most $\lfloor 6n/d_{02} \rfloor$, and try to find an irreducible polynomial $f_{02} = u_2^{d_{02}} H_{02}(x, -u_1/u_2) \in \mathrm{GF}(3)[x]$ of degree $6n$ such that $\gcd(u_2, f_{02}) = 1$, then u_2 is invertible modulo f_{02}. Then, there is a surjective homomorphism

$$\varPhi_{02} : \begin{cases} \mathrm{GF}(3)[x, y]/(H_{02}) \rightarrow \mathrm{GF}(3^{6n}) \cong \mathrm{GF}(3)[x]/(f_{02}) \\ \qquad\quad y \qquad\qquad \mapsto \qquad\quad -u_1/u_2, \end{cases}$$

where $H_{02}(x, y)$ holds $H_{02}(x, -u_1/u_2) \equiv 0 \pmod{f_{02}}$. In this polynomial selection, we need to modify Polynomial (2) to $su_2 - ru_1$. Note that r and s are chosen in $\mathrm{GF}(3)[x]$ of degree not larger than B_{02} in JL02-FFS, the size of the sieving area in the collection of relation step is

$$3^{B_{02}+1} \cdot 3^{B_{02}+1}. \tag{8}$$

From heuristic analysis in [16], JL02-FFS becomes optimized when we choose the smoothness bound B_{02} as

$$B_{02} = \lceil (4/9)^{1/3} (6n)^{1/3} \log_3 (6n)^{2/3} \rfloor. \tag{9}$$

and the extension degree d_{02} of $H_{02}(x, y)$ as $d_{02} = \lceil \sqrt{6n/(B_{02} + 1)} \rfloor$. For example, for $n = 97, 163, 193$, we have $(n, B_{02}) = (97, 21), (163, 26), (193, 28)$.

3.2 Polynomial Selection of JL06-FFS and Its Sieving Area

Next we describe an outline of the polynomial selection of JL06-FFS and estimate the size of the sieving area of JL06-FFS.

For each extension degree n of $\mathrm{GF}(3^6)$, we choose the smallest smoothness bound B_{06} in JL06-FFS satisfying the following condition,

$$(B_{06} + 1) \log(3^6) \geq \sqrt{n/B_{06}} \log(n/B_{06}) \tag{10}$$

For example, for $n = 97, 163, 193$, we have $(n, B_{06}) = (97, 3), (163, 4), (193, 4)$. Next, we choose positive integers d and d' such that $d \approx \sqrt{n/B_{06}}$ and $d' \approx \sqrt{nB_{06}}$, where $dd' \geq n$. After that, we randomly generate $g(y) \in \mathrm{GF}(3^6)[y]$ of degree d and set $H(x, y) = g(y) + x$. Finally, we try to find an irreducible polynomial f in $\mathrm{GF}(3^6)[x]$ of degree n, which divides $H(x, m)$, where $m \in \mathrm{GF}(3^6)[x]$ of degree d' is chosen randomly. In this polynomial selection, each of the leading coefficients of Polynomials (2) and (3) depends on r, so we avoid obtaining duplicate relations by fixing the leading coefficient of r as a monic polynomial. Therefore, the size of the sieving area in the collection of relations step is at most

$$(3^6)^{B_{06}+1} \cdot (3^6)^{B_{06}}. \tag{11}$$

3.3 Comparison of Sieving Area

We compare JL06-FFS with JL02-FFS with respect to the size of the sieving area in the collection of relations step in three classes of extension degree n: *experimental class* as {19, 31, 47, 61}, *medium-security class* as {97, 163, 193}, and *high-security class* as {239, 313, 353, 509}. Table 1 lists the smoothness bound and size of the sieving area in each variant. For each n, we obtain the smoothness bound B_{02} in Equation (9) and B_{06} in Equation (10), and estimate the size of the sieving area by Form (8) in JL02-FFS and by Form (11) in JL06-FFS.

Table 1. Parameters and sieving area

	n	Polynomial selection in JL02-FFS			Polynomial selection in JL06-FFS		
		$6n$	B_{02}	Size of sieving area	n	B_{06}	Size of sieving area
Experimental class	19	114	10	3.1×10^{10}	19	1	3.9×10^{8}
	31	186	12	2.5×10^{12}	31	2	2.1×10^{14}
	47	282	15	1.9×10^{15}	47	2	2.1×10^{14}
	61	366	17	1.5×10^{17}	61	2	2.1×10^{14}
Medium-security class	97	582	21	9.8×10^{20}	97	3	1.1×10^{20}
	163	978	26	5.8×10^{25}	163	4	5.8×10^{25}
	193	1158	28	4.7×10^{27}	193	4	5.8×10^{25}
High-security class	239	1434	30	3.8×10^{29}	239	4	5.8×10^{25}
	313	1878	34	2.5×10^{33}	313	5	3.1×10^{31}
	353	2118	36	2.0×10^{35}	353	5	3.1×10^{31}
	509	3054	42	1.1×10^{41}	509	6	1.6×10^{37}

Figure 1 shows the size of the required sieving area over $\mathrm{GF}(3^{6n})$. The sieving area in JL06-FFS is much smaller than that in JL02-FFS when $n \neq 31, 163$. Moreover, the differences between the sieving areas in JL06-FFS and in JL02-FFS increase along with the increase in n. The computational cost in the collection of relations step is closely related to the size of the sieving area, so the collection of relations step in JL06-FFS might be several times faster than that in JL02-FFS.

We have conducted experiments on the collection of relations step in JL02-FFS and JL06-FFS to confirm the difference between their computational costs of that step. Parameters in JL02-FFS and JL06-FFS are listed in Table 2. The

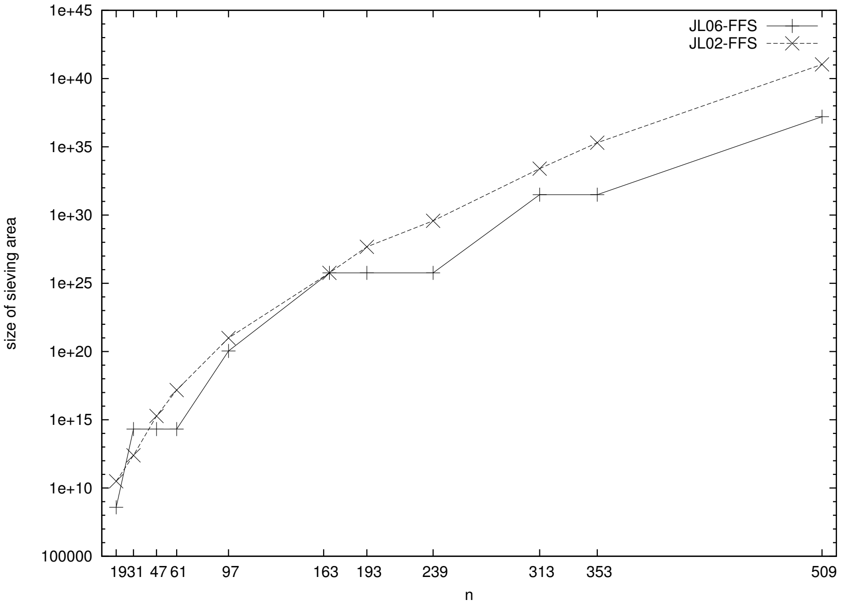

Fig. 1. Size of sieving area over GF(3^{6n}) in JL02-FFS and JL06-FFS

curves that we used in our experiments are superelliptic ones, but not C_{ab} curves as [12]. Note that we have only experimented with the experimental class as $n \in \{19, 31, 47, 61\}$, not with medium and high-security classes.

Table 2. Parameters in our experiments

n	Bit size of GF(3^{6n})	Experiments with JL02-FFS			Experiments with JL06-FFS		
		$6n$	B_{02}	$H_{02}(x, y)$	n	B_{06}	$H(x, y)$
19	181	114	10	$y^4 + x$	19	1	$y^5 + x$
31	295	186	12	$y^4 + x$	31	2	$y^4 + x$
47	447	282	15	$y^4 + x$	47	2	$y^5 + x$
61	581	366	17	$y^5 + x$	61	2	$y^6 + x$

In our experiments, we used 96 cores, each of which had the same performance about Intel 2.83GHz Xeon. We implemented the lattice sieve [26] in JL02-FFS as [12, 15, 16]. On the other hand, we implemented the polynomial sieve [10] in JL06-FFS, since we fixed r as a monic poynomial in the collection of relations step and so the lattice sieve might not be efficient. The details of our implementation in JL06-FFS are described in Section 4.

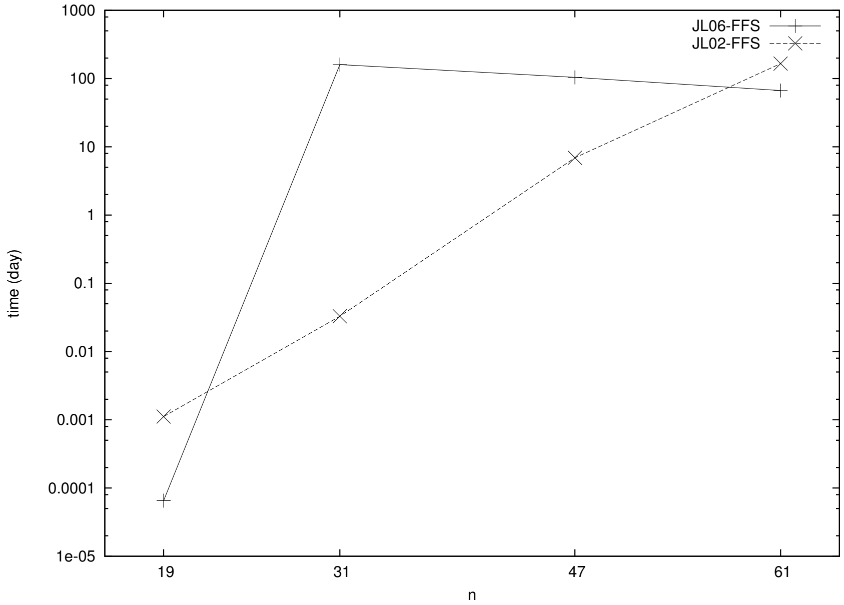

Fig. 2. Estimated time taken to compute entire sieving area in the collection of relations step over GF(3^{6n}) in JL02-FFS and JL06-FFS

Figure 2 shows the time complexity of JL02-FFS and JL06-FFS to compute the entire sieving area in the collection of relations step in GF(3^{6n}) with $n = 19$, 31, 47, 61, respectively. Note that we estimated the time when the computation lasts over one hour.

When $n = 19$, 61, our implementation on JL06-FFS is faster than that on JL02-FFS, and we confirm that JL06-FFS is more efficient than JL02-FFS for solving the DLP in GF(3^{6n}). In particular, when $n = 61$, our implementation of JL06-FFS takes about 66 days for the collection of relations step, but our implementation of JL02-FFS takes about 165 days for the same step. Therefore, the former is 2.5 times faster than the latter. Accordingly, we expect that JL06-FFS will be efficient for solving the DLP in GF(3^{6n}) for larger n.

4 Solving the DLP in GF($3^{6\cdot71}$)

In this section, we report that the DLP in GF($3^{6\cdot71}$) of 676-bit size is solved by improving JL06-FFS. In our implementation, we deal with four practical improvements, polynomial sieve, free relation, Galois action, and parallel Lanczos method.

Particularly, by using the polynomial $H(x, y) = y^6 + x$, we only need to find about 1/8 of the originally required relations in the collection of relations step. Furthermore, via the Galois action, the size of the matrix given by the relations is also decreased to 1/6 of the original. To the best of our knowledge, the 676-bit size is currently the record for solving the DLP in GF(3^{6n}).

4.1 Collection of Relations

In the collection of relations step, we collect many double smooth pairs $(r,\ s)$. The simple idea for collecting them is factoring Polynomials (2) and (3) for all pairs $(r,\ s)$. This is not practical since we have to factor them about $(3^6)^B \times (3^6)^{B+1}$ times. In order to reduce the number of factorings, we use a sieving method. The idea of sieving is merely factoring Polynomials (2) and (3) of the pair $(r,\ s)$, which has a high probability of becoming a double smooth pair. Such a pair is called a candidate.

The polynomial sieve [10] and the lattice sieve [26] are well-known sieving algorithms. Although the lattice sieve has been implemented in some experiments of the FFS [12, 15, 16], we implemented the polynomial sieve since r is fixed as a monic polynomial by the polynomial sieve in JL06-FFS, whereas neither r nor s is able to be fixed by the lattice sieve.

Polynomial Sieve We describe the polynomial sieve in Polynomial (2), namely, $rm+s$. Notice that we can also sieve in Polynomial (3) with the same procedure. Moreover, we discuss the case where s is fixed and omit the details when r is fixed. By fixed s, we can lead r such that $rm+s$ is divisible by $\mathfrak{p} \in B_R$ or its power, where the degree of $\mathfrak{p}$ is not larger than B. Additionally, $(rm+s)+k\mathfrak{p}$ with $k \in \mathrm{GF}(3^6)[x]$ is also divisible by $\mathfrak{p}$. Hence, we can obtain all r of degree less than or equal to B such that $rm+s$ is divisible by $\mathfrak{p}$. After computing such all r for each $\mathfrak{p}$, we can obtain the pair $(r,\ s)$ such that $rm+s$ is divisible by some $\mathfrak{p}$. If the summation of the degree of all $\mathfrak{p}$, which divide $rm+s$, reaches $\deg(rm+s)$, then $rm+s$ has a high probability of becoming B-smooth and the pair $(r,\ s)$ becomes a candidate.

In this procedure, the most time-consuming work is to compute $r+k\mathfrak{p}$ for all $k \in \mathrm{GF}(3^6)[x]$ whose degree is not larger than B. In characteristic two, Gordon and McCurley proposed a method using binary gray codes [10] to compute these $r+k\mathfrak{p}$. Using ternary gray codes, we can also compute them efficiently in characteristic three.

In the polynomial sieve, we sieve with all powers of $\mathfrak{p}$ whose degree is not larger than B. Since B is very small, such as 1 or 2 in our experiments, the power of $\mathfrak{p}$ is only $\mathfrak{p}^2$ when $\deg(\mathfrak{p}) = 1$. Such polynomials are exceptional since there are 3^6 monic irreducible polynomials of degree 1 in $\mathrm{GF}(3^6)[x]$. In this way, we can obtain only candidates each of which generates a relation in Equation (5) (except that r and s are not relatively prime). Thus, we only check the greatest common divisor of r and s, but not the smoothness of Polynomials (2) and (3) using the B-smooth test [10].

Free Relation By considering how a divisor $\langle\mathfrak{p}\rangle$ where $\mathfrak{p} \in B_R$ is factorized into divisors in $\mathrm{GF}(3^6)[x,\ y]/(H)$, namely, obtaining the following congruent expression that

$$H(x,\,y) \equiv \prod_{i=1}^{d}(y-t_i) \pmod{\mathfrak{p}},$$

where d is the degree of $H(x,\,y)$ on y, we can obtain a relation virtually for free, without the sieving procedure. We call such a relation a free relation.

The number of free relations depends on the degree d of $H(x,\,y)$ on y and the characteristic of the field treated in the FFS. In fact, there are about $\#B_A/d$ free relations in many cases and, furthermore, they increase when the characteristic is small. For example, in the case of $\mathrm{GF}(3^{6n})$ and $H(x,\,y) = y^6 + x$, there are about $\#B_A/2$ free relations since $y^6 + x$ is generally factored as $(y-t_1)^3(y-t_2)^3$ modulo $\mathfrak{p}$.

4.2 Linear Algebra

In the linear algebra step, we solve the linear equation depending on the relations. Specifically, we construct a matrix from the relations and reduce it to a much smaller one using the Galois action. After that, we solve the reduced linear equation modulo $(3^{6n}-1)/(3^6-1)$, by applying the parallel Lanczos method described as [3]. In this section, we describe the Galois action and our ideas about parallel computation of the matrix operation.

Galois Action Here, we consider to reduce unknowns of linear equations, using the Galois action which was presented in [18].

Let M' be the matrix given by the relations, whose row ${M'}_{(i)}$ means the i-th relation and j-th column ${M'}^{(j)}$ corresponds to the factorbase $\mathfrak{p}_j$. In order to use the Galois action, we choose the polynomial $f \in \mathrm{GF}(3^6)[x]$ satisfying that all coefficients of f are in $\mathrm{GF}(3)$ and $\deg(f) = n$, then we construct $\mathrm{GF}(3^{6n})$ as $\mathrm{GF}(3^6)[x]/(f)$. Let ϕ be the Frobenius power such that $\phi(\xi) = \xi^{3^n}$. As ϕ fixes the element x in $\mathrm{GF}(3)[x]/(f)$, we also have $\phi(x) = x$ in $\mathrm{GF}(3^6)[x]/(f)$ by the assumption of f. However, for an element $c \in \mathrm{GF}(3^6)\backslash\mathrm{GF}(3)$, ϕ does not fix c in $\mathrm{GF}(3^6)[x]/(f)$ by the above assumption that n is coprime to 6. The monic irreducible polynomial $\mathfrak{p}_j \in B_R$ of degree not larger than B, and we assume that $B = 1$ for convenience. In fact, $\mathfrak{p}_j = x + c_j$ where $c_j \in \mathrm{GF}(3^6)$ since $B = 1$, so we have

$$\phi(\mathfrak{p}_j) = \phi(x + c_j) = x + \phi(c_j)$$

in $\mathrm{GF}(3^6)[x]/(f)$. If c_j is not in $\mathrm{GF}(3)$, $c_j \neq \phi(c_j)$ in $\mathrm{GF}(3^6)[x]/(f)$. This fact implies that there are ordinarily many unknowns of linear equations, which can be rewritten by the other one via the Galois action. Clearly, for such $\mathfrak{p}_j$, there exists $\mathfrak{p}_{j'}$ satisfying that

$$\log_\gamma \mathfrak{p}_{j'} = \log_\gamma \phi(\mathfrak{p}_j) = 3^n \log_\gamma \mathfrak{p}_j \tag{12}$$

where $\mathfrak{p}_j \neq \mathfrak{p}_{j'}$. Therefore, we can remove the j'-th column ${M'}^{(j')}$ and set the j-th column ${M'}^{(j)}$ as ${M'}^{(j)} + 3^n {M'}^{(j')}$. Then we denote the new matrix M^* as the reduced M'. Notice that this technique is also used for the algebraic factorbase. Consequently, the number of unknowns is about 1/6 of the original; thus, the number of relations is reduced to about 1/6. In our implementation, we do not reduce the factorbase in the sieving phase (the computation is the same as the case without the Galois action). After sieving, we compress obtained relations using rewritable elements of the factorbase via the Galois action as Equation (12), and so the factorbase is reduced to about 1/6. Using this procedure, we almost do not lose the probability of obtaining the relation. Hence, this technique enables us to perform computations for the collection of relations step about 6 times as fast as before, and the linear algebra step can be also done about 6^2 times faster.

Parallel Lanczos method The reduced matrix M^* is reconstructed to optimize first, then we apply the parallel Lanczos method to it. Before explaining the reconstruction, we begin with the explanation of the parallel computation. Assume that there are four nodes written as $N_{1,1}, N_{1,2}, N_{2,1}, N_{2,2}$ and each node has 4 or 8 cores. As the Figure 3, we partition the reconstructed matrix M into four matrices $M_{i,j}$, and each $M_{i,j}$ is allotted to node $N_{i,j}$ respectively. The given vector $\boldsymbol{v}$ is also partitioned into $\boldsymbol{v}_1, \boldsymbol{v}_2$, and $\boldsymbol{v}_j$ is given to nodes $N_{i,j}$, $N_{i',j}$ where $i \neq i'$. Moreover, $M_{i,j}$ is partitioned into L matrices A_ℓ when $N_{i,j}$ has L cores.

$$M\boldsymbol{v} = \left(\begin{array}{c|c} M_{1,1} & M_{1,2} \\ \hline M_{2,1} & M_{2,2} \end{array}\right) \left(\begin{array}{c} \boldsymbol{v}_1 \\ \hline \boldsymbol{v}_2 \end{array}\right). \qquad M_{i,j}\boldsymbol{v}_j := A\boldsymbol{v}_j = \left(\begin{array}{c} A_1 \\ \hline A_2 \\ \hline \vdots \\ \hline A_L \end{array}\right) \boldsymbol{v}_j.$$

Fig. 3. Partitioning M into four matrices $M_{i,j}$ and $M_{i,j}$ into L matrices A_ℓ.

We now give the notation of the Lanczos method. The Lanczos method can operate only a symmetric matrix; however, the given matrix M is usually nonsymmetric. Therefore, we try to solve the linear equation of the form $M^T M\boldsymbol{v} = \boldsymbol{\alpha}$, where $\boldsymbol{v}$ is an unknown column vector consisting of the logarithms of the factorbase and $\boldsymbol{\alpha}$ is the given column vector. Note that computing $M^T M$ is not efficient, so we compute the vector $\boldsymbol{u} = M\boldsymbol{v}$ and $M^T\boldsymbol{u}$. For more details about this computation is in [22].

After partitioning M, we perform a parallel computation for $\boldsymbol{u} := M\boldsymbol{v}$ and $\boldsymbol{w} := M^T\boldsymbol{u}$ with $M_{i,j}$. Let $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, $\boldsymbol{u}_1$, and $\boldsymbol{u}_2$ be the partitioned vectors such that $\boldsymbol{v} = \boldsymbol{v}_1 \oplus \boldsymbol{v}_2$ and $\boldsymbol{u} = \boldsymbol{u}_1 \oplus \boldsymbol{u}_2$. From Algorithm 1, we obtain the partitioned vector $\boldsymbol{w}_i$ such that $\boldsymbol{w} = \boldsymbol{w}_i \oplus \boldsymbol{w}_{i'}$ in node $N_{i,j}$, where $i \in \{1,2\}$ and $i' = 3 - i$. The symbol j' also means that $j' = 3 - j$ for $j \in \{1,2\}$.
Lines 4, 5, and 6 describe the computation of $M^T u$. Note that in each node $N_{i,j}$, by regarding the column of $M_{i,j}$ as the row of $M_{j,i}^T$, we do not have to trade $M_{i,j}$ with $M_{j,i}^T$, namely, we can cut unnecessary operations.

Algorithm 1 (Computation with node $N_{i,j}$.)

```
Input : the partitioned matrix $M_{i,j}$ and the partitioned vector $\boldsymbol{v}_j$.
Output : the partitioned vector $\boldsymbol{w}_j$ such that $\boldsymbol{w}_1 \oplus \boldsymbol{w}_2 = M^T M\boldsymbol{v}$, where $j$ is equal to 1 or 2.
[Step for computation of $u := Mv$]
1. $\boldsymbol{u}_{i,j} := M_{i,j}\boldsymbol{v}_j$.
2. Give $\boldsymbol{u}_{i,j}$ to Node $N_{i,j'}$ and receive $\boldsymbol{u}_{i,j'}$ from $N_{i,j'}$.
3. $\boldsymbol{u}_i := \boldsymbol{u}_{i,j} + \boldsymbol{u}_{i,j'}$.
[Step for computation of $w := M^T u$]
4. $\boldsymbol{w}_{i,j} := M_{i,j}^T \boldsymbol{u}_i$.
5. Give $\boldsymbol{w}_{i,j}$ to Node $N_{i',j}$ and receive $\boldsymbol{w}_{i',j}$ from $N_{i',j}$.
6. $\boldsymbol{w}_j := \boldsymbol{w}_{i,j} + \boldsymbol{w}_{i',j}$.
```

We have discussed the parallel computations among nodes, and now we move on to the parallel computations among cores in one node. Here, A_ℓ denotes the partitioned matrix of $M_{i,j}$ such that $M_{i,j} = \oplus_{\ell=1}^{L} A_\ell$. From Algorithm 2, we can easily obtain $A_\ell \boldsymbol{v}_j$, and then we set the new vector $\boldsymbol{u}_{i,j} = (A_1\boldsymbol{v}_j, \ldots, A_L\boldsymbol{v}_j)^T$, where L is the number of cores in the same node. Similarly, we can easily obtain $A_\ell^T \boldsymbol{u}_i$ and compute $\boldsymbol{w}_{i,j} = \sum_{\ell=1}^{L} A_\ell^T \boldsymbol{u}_i$ by using Algorithm 3.

Algorithm 2 (Parallel computation of $M_{i,j}\boldsymbol{v}_j$ among L cores in the same node.)

```
Input : the partitioned matrix $A := M_{i,j}$ whose size is $s \times t$ and the partitioned $t$-vector $\boldsymbol{v}_j$.
Output : the partitioned vector $\boldsymbol{u}_{i,j}$ such that $\boldsymbol{u}_{i,j} = A\boldsymbol{v}_j$.
1. Compute $\boldsymbol{b}_\ell := A_\ell \boldsymbol{v}_j$ for $\ell = 1$ to $\ell = L$ in parallel.
2. $\boldsymbol{u}_{i,j} = \oplus_{\ell=1}^{L} \boldsymbol{b}_\ell$.
```

Algorithm 3 (Parallel computation of $M_{i,j}^T \boldsymbol{u}_i$ among L cores in the same node.)

```
Input : the partitioned matrix $A := M_{i,j}$ whose size is $s \times t$ and the partitioned $s$-vector $\boldsymbol{u}_i$.
Output : the partitioned vector $\boldsymbol{w}_{i,j}$ such that $\boldsymbol{w}_{i,j} = A^T \boldsymbol{u}_i$.
1. Compute $\boldsymbol{c}_\ell := A_\ell^T \boldsymbol{u}_i$ for $\ell = 1$ to $\ell = L$ in parallel.
2. $\boldsymbol{w}_{i,j} = \sum_{\ell=1}^{L} \boldsymbol{c}_\ell$.
```

From the parallel computations of $M_{i,j}\boldsymbol{v}_j$ and so on, we obtain the vector $M^T M\boldsymbol{v}$ from Algorithm 1 and 2. Therefore, we need to reconstruct M so that each node has the balanced calculation amount of computing $M_{i,j}\boldsymbol{v}_j$ and so on. It is clear that the calculation amount depends on the number of non-zero elements in the allotted matrix, and the distribution of non-zero elements in M is not uniformity. In fact, the number of non-zero elements in a column of M is not balanced, but that in a row is balanced. Thus, we reconstruct the new matrix M so that the number of non-zero elements in $M_{1,1}$ and $M_{2,1}$ is almost equal to that in $M_{1,2}$ and $M_{2,2}$ by sorting columns of M^* defined in the section of the Galois action. We perform a similar strategy as above for the parallel computation among cores in the same node, namely, A is partitioned into 4 or

8 smaller matrices A_ℓ so that each A_ℓ has almost the same number of non-zero elements.

4.3 Computation Results

In this section, we describe our computation results of the 676-bit DLP in GF($3^{6\cdot 71}$), which contains a multiplicative subgroup whose order is a 112-bit prime. We construct GF(3^6) as GF(3)$[z]/(z^6 + 2z + 2)$ and define a mapping $\psi : \mathbb{Z} \to \mathrm{GF}(3^6)[x]$, such that $\psi^{-1} : z \mapsto 3, x \mapsto 3^6$, in order to represent the element in GF(3^6)$[x]$.

In the polynomial selection step, we set $H(x, y) = y^6 + x$ in order to use the Galois action. Moreover, we select $m \in \mathrm{GF}(3^6)[x]$ such that all its coefficients are in GF(3) to construct f whose coefficients are also in GF(3). By an easy computation, we obtain proper m and f as follows,

$$\begin{aligned} m &= \psi\,(\texttt{0x456bc 60e76c11 1e679735 c929fc55}) \\ f &= \psi\,(\quad\;\; \texttt{0x9 2d3e5daf 5ac01130 4e6909f7 09cc8833 baa757d3} \\ &\quad\;\; \texttt{17dc6f99 9c8b98b5 ab8baa01 d68ec151 aec39e2e ed081c79} \\ &\quad\;\; \texttt{d851066b 3ffb2a4f a3e19c1e cef46675 0918a26d 9c7cacd4} \\ &\quad\;\; \texttt{8d74ccfe 2c1d3b79 e81e6138 ab06aef4}). \end{aligned}$$

Then, GF(3^{6n}) is constructed as GF(3^6)$[x]/(f)$. When we set the smoothness bound $B = 2$, there are 266,085 elements in the rational factorbase and 265,721 elements in the algebraic factorbase, so we need to collect at least 531,806 relations. However, the size of the sieving area when $B = 2$ is too small to collect enough relations.

We settle this problem by using the Galois action, since we can considerably reduce the number of required elements in the factorbase described in Section 4.2. In fact, we need only 88,674 relations, and so this number is about 1/6 the number of the originally required relations.

Moreover, we deal with free relations which are obtained without sieving. If we choose $H(x, y)$ as $y^6 + x$, then it is fortunately factored as $(y - t_1)^3(y - t_2)^3 \pmod{\mathfrak{p}}$ for most of elements $\mathfrak{p}$ in the factorbase, and so there are 132,860 ($\approx \#B_A/2$) free relations. Even if we delete many duplicates which are produced by using the Galois action, 22,155 free relations remain. Thus, we only have to find at least 66,519 relations in the collection of relations step, and this number is about 1/8 that of the originally required relations.

In the collection of relations step, we use the polynomial sieve described in Section 4.1 and compute relations using five nodes, each consisting of Intel Quad-Core Xeon E5440 (2.83 GHz) × 2 CPUs with 16-GB RAM, one node consisting of Intel Quad-Core Xeon X5355 (2.66 GHz) × 2 CPUs with 16-GB RAM, and twelve nodes, each consisting of Intel Quad-Core Xeon L5420 (2.33 GHz) × 1 CPU with 4-GB RAM, total of 96 cores. In 18 days of computation, after removing duplicates, we found 66,646 relations. Thus, we obtained a total of 88,801 relations, which are enough to solve the linear equation in Equation (7).

The linear equation constructed from the relations has to be solved modulo $(3^{6\cdot 71}-1)/(3^6-1)$; however, the Lanczos method may fail when the modulus has a small prime factor. Therefore, we work modulo the factor N_i of $(3^{6\cdot 71}-1)/(3^6-1)$,

$$\begin{aligned}
N_1 &= (3^{2\cdot 71}+3^{71}+1)/(13\cdot 5113),\\
N_2 &= (3^{2\cdot 71}-3^{71}+1)/(7\cdot 210019\cdot 49682251\cdot 55126531),\\
N_3 &= (3^{71}+1)/(2^2\cdot 853\cdot 2131\cdot 82219),\\
N_4 &= (3^{71}-1)/2.
\end{aligned}$$

where every prime factor of N_i is larger than 30 bits and N_i is relatively prime to each other.

We use a cluster with four nodes, each consisting of Intel Quad-Core Xeon E5440 (2.83 GHz) × 2 CPUs with 16-GB RAM, and three clusters with four nodes, each consisting of Intel Quad-Core Xeon L5420 (2.33 GHz) × 1 CPU with 4-GB RAM. With about 12 hours computation, we solve the linear equation modulo N_i via the parallel Lanczos method with the four nodes described in Section 4.2 on each cluster. With the Chinese remainder theorem and the Galois action of ϕ, we solved discrete logarithms of the elements in the factorbase modulo $N=\prod_{i=1}^{4} N_i$.

In the individual logarithm step, our target of computing the logarithm is the element

$$\begin{aligned}
\pi(x) &= \psi(\lfloor \pi\times 10^{202}\rfloor)\\
&= (z^4+z^3+2z^2+1)x^{70}+\cdots+(z^5+2z^4+2z^3+z^2+2)
\end{aligned}$$

in basis $\gamma=\psi(\mathtt{0x456})$. We choose the representation of $\pi(x)$ as a product of elements of degree at most 7 as follows:

$$\begin{aligned}
&\gamma^{\tau}\pi(x) \equiv z_1/z_2 \pmod{f},\ \textit{where}\\
z_1 &= \psi(\mathtt{0x333})\times\psi(\mathtt{0x345})\times\psi(\mathtt{0x427})\times\psi(\mathtt{0x43b})\times\psi(\mathtt{0x4c3})\\
&\quad\times\psi(\mathtt{0xd909\ 66c7e3ec})\times\psi(\mathtt{0x293996d\ cc380672})\\
&\quad\times\psi(\mathtt{0x3ff378e\ 3d4659d0})\times\psi(\mathtt{0x6\ 27d6c281\ 0a0fc5a2})\\
&\quad\times\psi(\mathtt{0x8\ f4797e29\ a9ec3b4a}),\\
z_2 &= \psi(\mathtt{0x318})\times\psi(\mathtt{0x45\ 4c6fbfd4})\times\psi(\mathtt{0x54\ c69e6f97})\\
&\quad\times\psi(\mathtt{0x1686d\ 42782189})\times\psi(\mathtt{0x3cf67a5\ 84055cd8})\\
&\quad\times\psi(\mathtt{0x8\ f68ab2e2\ 5d2bc04f})\times\psi(\mathtt{0xb\ cc56922c\ f651b383}),\\
\tau &= \qquad \mathtt{0x2\ 0f822e8c\ ac48792a\ e2aea337\ c9002b49\ bbf1b864}\\
&\quad\mathtt{43a6111b\ 24c5593d\ e44daf43\ e26de26e\ 1f85f982\ 1ba485b3}\\
&\quad\mathtt{beda74bd\ f782626d\ 6cd38bb2\ 8f829867\ 5dc04adc\ f8741c24},
\end{aligned}$$

and z_1, z_2 are 7-smooth. Then, we compute the logarithms of z_1 and z_2 in basis γ using the special-$\mathtt{q}$ descent technique [16, 18]. With about 14 days computation using five nodes, each consisting of Intel Quad-Core Xeon E5440 (2.83 GHz) ×

2 CPUs with 16-GB RAM, and one node consisting of Intel Quad-Core Xeon X5355 (2.66 GHz) × 2 CPUs with 16-GB RAM, we compute the logarithms,

$$\begin{aligned}
\log_\gamma z_1 \equiv & \quad \texttt{0x3 fc71c577 10be8e3f e7af0fba e00e711f 0ad6dd50} \\
& \texttt{38fb8f26 c0fadb3b 448cab2f 67671247 285f9e95 dc501717} \\
& \texttt{d9def844 a75f9e58 f04a9bd2 3a5d0fdb 8f8ebb9f fea4deea}, \\
\log_\gamma z_2 \equiv & \quad \texttt{0x4 82febaec ae4382e0 e651f577 09df4e7d 99d99d34} \\
& \texttt{03db5d5e 521c4e2b da89ec33 6c9d45d6 2dd1f982 2f198fb2} \\
& \texttt{6c069414 3b0b1544 ece8e4b1 5304872f 6ff261fd 03b271c7}.
\end{aligned}$$

modulo N, and so we obtain $\log_\gamma \pi(x) \bmod N$.

The logarithm in multiplicative subgroups of less than 30 bits are computed using the Pollard's ρ method in a minute. Using the Pohlig-Hellman method, we compute the logarithm $\log_\gamma \pi(x)$:

$$\begin{aligned}
\log_\gamma \pi(x) = & \quad \texttt{0x8 78b54797 2fb6ff9b 57add5d5 11f69de6 a3853f98} \\
& \texttt{68d53cc0 5b531076 2872ac6a 320874bf ba6d66d6 8e5e245f} \\
& \texttt{39778f02 31ae791a acbab8c7 5ee6850c 9f5df0e5 f6b8ab0b} \\
& \texttt{95d8bdb1 aea95b1f bad82465 25590f66}
\end{aligned}$$

and completely solve the DLP in GF($3^{6 \cdot 71}$) of 676-bit.

4.4 For Larger Extension Degrees

We have solved the DLP in GF(3^{6n}) for n in the experimental class, where the smoothness bound B (i.e., B_{06}) is less than or equal to 2 (ref. Table 1). Note that the size of the sieving area increases $(3^6)^2$-fold if the smoothness bound B increases by one (see Form (11)). However, we expect that, if we set $B = 3$, the DLP in GF($3^{6 \cdot 97}$) might be computed for several years by using dozens of our computational resources through various techniques such as large prime variation, block sieving and sieving via bucket sort [29, 4], and SIMD implementation.

5 Concluding Remarks

In this study, we implemented a new variant of the FFS in GF(3^{6n}) (n is a prime), proposed by Joux and Lercier in 2006 [18], and compared it with the earlier variant, which was also proposed by Joux and Lercier in 2002 [16] with practical experiments. In solving the DLP in GF(3^{6n}), these two variants of the FFS have the same asymptotic complexity, but we expected the new variant to be more efficient than the earlier one in some extension degrees n. From our experimental results, we confirmed this forecast when the extension degree $n = 19, 61$. Moreover, with our implementations, we succeeded in solving the DLP in GF($3^{6 \cdot 71}$) of 676-bit size with about 33 days computation.

Table 3. Records for solving the DLP in finite fields

Finite Fields	GF(p)	GF(2^n)	GF(p^3)	GF(p^{30})	**GF(3^{6n})**
Reference Date Algorithm	[21] Feb. 5, 2007 NFS*	[15] Sep. 22, 2005 JL02-FFS	[20] Aug. 23, 2006 JLSV06-NFS†	[18] Nov. 9, 2005 JL06-FFS-2‡	**This Work** **Dec. 9, 2009** **JL06-FFS**
Collection of Relations	Many CPUs¶	4 nodes of 16 Itanium 2 (1.3GHz)	16 Alpha processors (1.15GHz)	16 Alpha processors (1.15GHz)	**Xeon (2.83GHz) 96 cores**
Linear Algebra	12–24 Xeon (3.2GHz)	4 nodes of 16 Itanium 2 (1.3GHz)	16 Alpha processors (1.15GHz)	16 Alpha processors (1.15GHz)	**Xeon (2.83GHz) 80 cores**
Timing Bit Size	33 days 532	17 days 613	19 days 394	12 hours 556	**33 days** **676**

*NFS: Number Field Sieve [9, 17]. †JLSV06-NFS: NFS in the medium prime case [20].
‡See footnote 2 on page 2. ¶There are no detailed descriptions of computational resources in [21].

We have experimented with the DLP in GF(3^{6n}) required for pairing-based cryptosystems. The security of pairing-based cryptosystems relies on the difficulty of the DLP in various finite fields, for example, GF(2^{4n}) and GF(p^{12}). Table 3 presents the current records for solving the DLP in various finite fields. All the DLPs used for pairing-based cryptosystems have not examined yet. It is an open problem to analyze the hardness of the DLP with practical key sizes in such finite fields.

References

1. L. M. Adleman. The function field sieve. *ANTS-I*, LNCS 877, pp. 108–121, 1994.
2. L. M. Adleman and M.-D. A. Huang. Function field sieve method for discrete logarithms over finite fields. *Inform. and Comput.*, Vol. 151, pp. 5–16, 1999.
3. K. Aoki, T. Shimoyama, and H. Ueda. Experiments on the linear algebra step in the number field sieve. *IWSEC 2007*, LNCS 4752, pp. 58–73, 2007.
4. K. Aoki and H. Ueda. Sieving using bucket sort. *ASIACRYPT 2004*, LNCS 3329, pp. 92–102, 2004.
5. P. S. L. M. Barreto, S. Galbraith, C. Ó hÉigeartaigh, and M. Scott. Efficient pairing computation on supersingular abelian varieties. *Des., Codes Cryptogr.*, Vol. 42, No. 3, pp. 239–271, 2007.
6. J.-L. Beuchat, N. Brisebarre, J. Detrey, E. Okamoto, M. Shirase, and T. Takagi. Algorithms and arithmetic operators for computing the η_T pairing in characteristic three. *IEEE Trans. Comput.*, Vol. 57, No. 11, pp. 1454–1468, 2008.
7. D. Boneh, D. Crescenzo, R. Ostrovsky and G. Persiano. Public key encryption with keyword search. *EUROCRYPT 2004*, LNCS 3027, pp. 506–522, 2004.
8. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. *SIAM J. Comput.*, Vol. 32, No. 3, pp. 586–615, 2003.
9. D. M. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM J. Discrete Math.*, vol. 6, no. 1, pp. 124-138, 1993.
10. D. M. Gordon and K. S. McCurley. Massively parallel computation of discrete logarithms. *CRYPTO' 92*, LNCS 740, pp. 312–323, 1992.

11. R. Granger. Estimates for discrete logarithm computations in finite fields of small characteristic. *Cryptography and Coding 2003*, LNCS 2898, pp. 190–206, 2003.
12. R. Granger, A. J. Holt, D. Page, N. P. Smart, and F. Vercauteren. Function field sieve in characteristic three. *ANTS-VI*, LNCS 3076, pp. 223–234, 2004.
13. R. Granger, D. Page, and M. Stam. Hardware and software normal basis arithmetic for pairing-based cryptography in characteristic three. *IEEE Trans. Comput.*, Vol. 54, No. 7, pp. 852–860, 2005.
14. D. Hankerson, A. Menezes, and M. Scott. Software implementation of pairings. In *Identity-Based Cryptography*, pp. 188–206, 2009.
15. A. Joux et al. Discrete logarithms in GF(2^{607}) and GF(2^{613}). Posting to the Number Theory List, available at `http://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind0509&L=nmbrthry&T=0&P=3690`, 2005.
16. A. Joux and R. Lercier. The function field sieve is quite special. *ANTS-V*, LNCS 2369, pp. 431–445, 2002.
17. A. Joux and R. Lercier. Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the Gaussian integer method. *Math. Comp.*, Vol. 72, No. 242, pp. 953–967, 2002.
18. A. Joux and R. Lercier. The function field sieve in the medium prime case. *EUROCRYPT 2006*, LNCS 4004, pp. 254–270, 2006.
19. A. Joux, R. Lercier, D. Naccache, and E. Thome. Oracle-assisted static Diffie-Hellman is easier than discrete logarithms. *Cryptography and Coding 2009*, LNCS 5921, pp. 351–367, 2009.
20. A. Joux, R. Lercier, N. P. Smart, and F. Vercauteren. The number field sieve in the medium prime case. *CRYPTO 2006*, LNCS 4117, pp. 326–344, 2006.
21. T. Kleinjung et al. Discrete logarithms in GF(p) - 160 digits. Posting to the Number Theory List, available at `http://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind0702&L=nmbrthry&T=0&P=194`, 2007.
22. B. A. LaMacchia and A. M. Odlyzko. Solving large sparse linear systems over finite fields. *CRYPTO' 90*, LNCS 537, pp. 109–133, 1991.
23. R. Matsumoto. Using C_{ab} curves in the function field sieve. *IEICE Trans. Fundamentals*, Vol. E82-A, pp. 551–552, 1999.
24. A. J. Menezes, T. Okamoto and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Inform. Theory*, Vol. 39, No. 5, pp. 1639–1646, 1993.
25. D. Page, N. P. Smart, and F. Vercauteren. A comparison of MNT curves and supersingular curves. *Appl. Algebra Engrg. Comm. Comput.*, Vol. 17, No. 5, pp. 379–392, 2006.
26. J. Pollard. The lattice sieve. In *The Development of the Number Field Sieve*, pp. 43–49, 1991.
27. C. Pomerance and J. W. Smith. Reduction of huge, sparse matrices over finite fields via created catastrophes. *Experiment. Math.*, Vol. 1, No. 2, pp. 89–94, 1992.
28. O. Schirokauer. The special function field sieve. *SIAM J. Discrete Math.*, Vol. 16, No. 1, pp. 81–98, 2003.
29. G. Wambach and H. Wettig. Block sieving algorithms. Technical Report 190, Informatik, Universität zu Köln, 1995.
30. D. H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Inform. Theory*, Vol. 32, No. 1, pp. 54–62, 1986.