Experimental Analysis of Cheon’s Algorithm against Pairing-Friendly Curves

Tetsuya Izu
FUJITSU LABORATORIES LTD.
4-1-1 Kamikodanaka, Nakahara-ku
Kawasaki, 211-8588, Japan
Email: izu@labs.fujitsu.com

Masahiko Takenaka
FUJITSU LABORATORIES LTD.
4-1-1 Kamikodanaka, Nakahara-ku
Kawasaki, 211-8588, Japan
Email: takenaka@labs.fujitsu.com

Masaya Yasuda
FUJITSU LABORATORIES LTD.
4-1-1 Kamikodanaka, Nakahara-ku
Kawasaki, 211-8588, Japan
Email: myasuda@labs.fujitsu.com

Abstract—The discrete logarithm problem (DLP) is one of the familiar problem on which some cryptographic schemes rely. In 2006, Cheon proposed an algorithm for solving DLP with auxiliary input which works better than conventional algorithms.

In this paper, we show our experimental results of Cheon’s algorithm on a pairing-friendly elliptic curve defined over GF(3127). It is shown that the algorithm combined with the kangaroo method has an advantage over that combined with the baby-step giant-step method in the sense that the required time and space are smaller.

Then, for the algorithm combined with the kangaroo-method, speeding-up techniques are introduced. Based on our experimental results and the speeding-up techniques, we evaluate the required time and space for some pairing-friendly elliptic curves. As results, a portion of pairing-friendly elliptic curves can be analyzed by Cheon’s algorithm at reasonable cost.

Keywords—Discrete logarithm problem (DLP); Cheon’s algorithm; pairing-friendly elliptic curve on GF(3127); experimental results

I. INTRODUCTION

Let G be an additive group generated by G with order r (prime), namely, G = ⟨G⟩. (Typically, G is the Mordell-Weil group on an elliptic curve defined over a finite field.) Finding α ∈ Z/rZ on input G, αG ∈ G is called the discrete logarithm problem (DLP). In the general setting, the most efficient algorithms for solving DLP require O(√r) in time, and DLP is considered to be infeasible if parameters are properly chosen. Some cryptographic schemes rely on this infeasibility to assure the security.

In 2006, Cheon proposed an algorithm for solving DLP with auxiliary input (DLPwAI), namely, finding α ∈ Z/rZ on input G, αG and αdG for a positive integer d dividing r − 1 [7]. Since the time complexity of Cheon’s algorithm is O(log r √(r − 1)/d + √d) and the space complexity is O(max (√(r − 1)/d, √d)). Cheon’s algorithm may work better than conventional algorithms. Especially, when d ≈ √r, it only requires O(√r) in time and space.

Recently, new (possibly infeasible) problems related to DLP are introduced to assure the security of some cryptographic schemes such as ℓ-WDH problem [14], ℓ-SDH problem [3], ℓ-sSDH problem [5], ℓ-BDHI problem [4] and ℓ-BDHE problem [5]. A common property of these problems is that G, αG, α2G, …, αℓG can be used to solve these problems. When ℓ is larger than d, Cheon’s algorithm is able to solve these problems, and thus, such parameters should be avoided in cryptographic use.

These cryptographic schemes involve pairing operations. In order to speed-up the pairing operations, particular elliptic curves called “pairing-friendly curves” are widely used. A portion of pairing-friendly curves have characteristic that r−1 is smooth, where r is the order of elements. Consequently, Cheon’s algorithm can be applied to such pairing-friendly elliptic curves, and the above problems can be solved.

When Cheon’s algorithm is implemented, there are two choices, namely, the kangaroo method and the baby-step giant-step (BSGS) method [7], [8]. In general, the kangaroo method requires less space compared to the baby-step giant-step method. However, a detailed comparison of these methods in actual experiments has not been reported so far. Note that the only experimental result of Cheon algorithm uses the baby-step giant-step method [9].

In this paper, we implemented Cheon’s algorithm with both methods. As far as the authors know, this is the first implementational result of Cheon’s algorithm combined with the kangaroo method. Then, we compare both methods from the viewpoint of time and space. Both in time and space, the kangaroo method works better than the baby-step giant-step method.

Then, for Cheon’s algorithm combined with the kangaroo method, speeding-up techniques are introduced. Based on our experimental results and the speeding-up techniques, we evaluate the required time and space for some pairing-friendly elliptic curves. As results, a portion of pairing-friendly elliptic curves can be analyzed by Cheon’s algorithm at reasonable cost.

The rest of this paper is organized as follows: in section II, we introduce the baby-step giant-step algorithm [16] and its combination with Cheon’s algorithm. In section III, the kangaroo method [15] and its combination are introduced. Then experimental results of Cheon’s algorithm combined with the baby-step giant-step method and the kangaroo method are shown in section IV and V. In section VI, speeding-up techniques are described. And finally, section
VII evaluates time and space prediction to analyze against some pairing-friendly elliptic curves.

II. CHEON’S ALGORITHM AND BSGS METHOD

This section briefly explains Shanks’ baby-step giant-step (BSGS) method [16] and its combination with Cheon’s algorithm [7], [8].

Let $G = \langle G \rangle$ be an additive group generated by $G$ with order $r$ (prime). The discrete logarithm problem (DLP) in $G$ is to find $\alpha \in \mathbb{Z}/r\mathbb{Z}$ on input $G \in G$ and $\alpha G \in G$. In the general setting, the most efficient algorithms for solving DLP require $O(\sqrt{r})$ in time. In fact, Shanks’ BSGS method [16] requires $O(\sqrt{r})$ group operations in time and $O(\sqrt{r})$ group elements in space.

A. Baby-step Giant-step Method

In 1971, Shanks proposed an algorithm for solving DLP [16], which is called as the baby-step giant-step (BSGS) method. Instead of finding $\alpha$ directly, BSGS method searches two integers $i, j$ such that $\alpha = i + j \sqrt{r}$ and $0 \leq i, j < m = \lceil \sqrt{r} \rceil$ (where $\lceil x \rceil$ is the ceiling function of $x$). Such $i, j$ are uniquely determined. Since $\alpha G = (i + j \sqrt{r})G = iG + jG'$ for $G' = mG$, we have a relation $G_1 - iG = jG'$ for $G_1 = \alpha G$.

BSGS method consists of two steps: in the 1st step (the baby-step), we compute

$$G_1, G_1 - G, G_1 - 2G, \ldots, G_1 - (m - 1)G$$

successively and store them in a table. In the 2nd step (the giant-step), we compute

$$G', 2G', \ldots, (m - 1)G'$$

successively and store them in another table. Then, we search a collision $G_1 - iG = jG'$ between these tables and thus a solution $\alpha = i + j \sqrt{r}$ is obtained. Since $O(m)$ group operations and $O(m)$ group elements are required in both steps, the time and space complexity of BSGS method are $O(\sqrt{r})$ group operations and $O(\sqrt{r})$ group elements, respectively.

B. Cheon’s Algorithm

In 2006, Cheon proposed an algorithm for solving DLP with auxiliary input [7], which finds $\alpha$ on input $G$, $G_1 = \alpha G$ and $G_d = \alpha^d G$ for an integer $d$ dividing $r - 1$.

Let us describe Cheon’s algorithm. A goal of Cheon’s algorithm is to find an integer $k \in \mathbb{Z}/r\mathbb{Z}$ such that $\alpha = \zeta^k$ for a generator of the multiplicative group $\zeta \in (\mathbb{Z}/r\mathbb{Z})^*$. Such $k$ is uniquely determined. To do so, Cheon’s algorithm finds two integers $k_1, k_2$ such that $k = k_1 + k_2(r - 1)/d$ satisfying $0 \leq k_1 < (r - 1)/d$, $0 \leq k_2 < d$ in the following two steps. An outline of Cheon’s algorithm is shown in Algorithm 1.

1 Note that the generator $\zeta$ can be found in an efficient way.

---

Algorithm 1. Cheon’s Algorithm with the BSGS method

Input: $G$, $G_1 = \alpha G$, $G_d = \alpha^d G \in G$
Output: $\alpha \in \mathbb{Z}/r\mathbb{Z}$

1. Find a generator $\zeta \in (\mathbb{Z}/r\mathbb{Z})^*$
2. $\zeta_d \leftarrow \zeta^d$, $d' \leftarrow (r - 1)/d$
3. [Step 1] $d_1 \leftarrow \lceil \sqrt{d} \rceil$
4. Find $0 \leq u_1, v_1 < d_1$ such that $\zeta_d^{-u_1} G_d = \zeta_d^{v_1} G$
5. $k_1 \leftarrow u_1 + v_1 d_1$
6. [Step 2] $d_2 \leftarrow \lceil \sqrt{d} \rceil$
7. Find $0 \leq u_2, v_2 < d_2$ such that $\zeta^{-u_2 d'} G_1 = \zeta^{u_2 + v_2 d'} G$
8. $k_2 \leftarrow u_2 + v_2 d_2$
9. Output $\alpha = \zeta^{k_1 + k_2 d'}$

Step 1 searches the integer $k_1$ such that $\alpha^d = \zeta^{k_1}$ ($\zeta_d = \zeta^d$), or equivalently, searches two integers $u_1, v_1$ such that $\alpha^d \zeta_d^{-u_1} = \zeta^{v_1} d_1$
satisfying

$$0 \leq u_1, v_1 < d_1 = \left\lceil \frac{\sqrt{(r - 1)/d}}{d} \right\rceil .$$

Such $u_1, v_1$ are uniquely determined. In practice, it searches $u_1, v_1$ such that $\zeta_d^{-u_1} G_d = \zeta_d^{v_1} d_1 G$. Step 2 searches the integer $k_2$ such that $\alpha = \zeta^{k_1 + k_2(r - 1)/d}$ in the similar way, or equivalently, searches integers $u_2, v_2$ such that

$$\alpha \zeta_d^{-u_2 (r - 1)/d} = \zeta^{u_2 + v_2 d_2 (r - 1)/d}$$

satisfying

$$0 \leq u_2, v_2 < d_2 = \left\lceil \sqrt{d} \right\rceil .$$

Such $u_2, v_2$ are uniquely determined. In practice, it searches $u_2, v_2$ such that $\zeta^{-u_2 (r - 1)/d} G_1 = \zeta^{u_2 + v_2 d_2 (r - 1)/d} G_1$. In Cheon’s algorithm, searching $u_1, v_1$ in Step 1 and searching $u_2, v_2$ in Step 2 can be combined with the BSGS method as a subroutine.

C. Complexity

In [7], [8], the time and space complexity are dependent not only $r$ but also $d$:

- Time : $O \left( \log r \cdot \left( \sqrt{(r - 1)/d} + \sqrt{d} \right) \right)$ (group operations),
- Space : $O \left( \max \left( \sqrt{(r - 1)/d}, \sqrt{d} \right) \right)$ (group elements).

Here, term $\log r$ in time complexity is complexity for scalar multiplication on group $G$.

Kozaki, Kutsuna and Matsuo introduced the precomputation table on fixed point to Cheon’s algorithm (KKM method) [13]. This method reduced the time complexity to $O \left( \sqrt{(r - 1)/d} + \sqrt{d} \right)$ group operations.
III. CHEON’S ALGORITHM AND KANGAROO METHOD

In [7], [8], Cheon also proposed a simple extension from a combination with the BSGS method to one with the kangaroo method [15]. Pollard’s kangaroo method is one of the most efficient algorithms for solving DLP in the general setting. (Please see [15] about the kangaroo method in detail.) Cheon’s algorithm combined with the kangaroo method reduces the table size compared with that with the BSGS method. In fact, the distinguished point technique (described later) reduces it. In order to apply the kangaroo method into Cheon’s algorithm, three techniques are needed; the search-space expansion, a random-walk function, and the distinguished point technique. This section shows them in detail.

A. Search-space Expansion

In the way of step 1 in subsection II-B, the search space is divided by \( d_1 \). Therefore, the size of search space is \( O(\sqrt{(r-1)/d}) \). In this search space, \( u_1, v_1 \) are uniquely determined. Thus the search space is completely searched, and all searched values should be stored.

On the other hand, the search space is not divided in the way in the kangaroo method. In this way, step 1 is modified by the following way: search \( u_1, v_1 \) such that

\[
\alpha^{d_u u_1} = \zeta^{v_1} d_v
\]

satisfying

\[
0 \leq u_1, v_1 < \lceil(r-1)/d \rceil.
\]

The size of the total search space is \( O((r-1)/d)^2 \).\footnote{The size of this search space is \( O((r-1)/d)^2 \).} In this search space, however, there are \( O((r-1)/d) \) pairs of \( u_1, v_1 \), and the search space for one pair of \( u_1, v_1 \) is \( O((r-1)/d) \) on average. By using following random-walk function, substantive size of search space is expected to be \( O((r-1)/d) \) by the birthday paradox.

This expansion can be similarly applied to step 2, however, the details are omitted in this paper.

B. Using a Random-walk Function

In order to reduce the search space by the birthday paradox, elements must be randomly chosen. Consequently, a random-walk function is introduced \[7\], \[8\]. Requirements of random-walk function for the kangaroo method are as follows: an element is chosen pseudo-randomly, and an element can be calculated with information of previously chosen elements. Additionally, an assumption is required in Cheon’s algorithm, that is to describe a chosen element as \( \zeta^2 G \).

Algorithm 2. Cheon’s Algorithm based on kangaroo method

\begin{itemize}
\item Input: \( G, G_1 = \alpha G, d_1 = w_1 G \in \mathbb{G} \)
\item Output: \( \alpha \in \mathbb{Z}/r\mathbb{Z} \)
\end{itemize}

1. Find a generator \( \zeta \in \mathbb{Z}/r\mathbb{Z}^* \)
2. \( \zeta_d = \zeta^d, \zeta_d' = \zeta^d, d' \leftarrow (r-1)/d \)
3. [Step 1] \( F_{d_1}^{(0)}(G_d) = \zeta_d G_d, F_{d_1}^{(0)}(G') = \zeta_d' G' \)
4. Find 0 \( \leq i_1, j_1 \) such that \( F_{d_1}^{(i_1)}(G_{d_1}) = F_{d_1}^{(j_1)}(G) \)
5. \( u_1 \leftarrow \sum_{i=0}^{i_1} f_i F_{d_1}^{(i)}(G_{d_1}) \mod d' \)
6. \( v_1 \leftarrow \sum_{i=0}^{i_1} f_i F_{d_1}^{(i)}(G) \mod d' \)
7. [Step 2] \( F_{d_1}^{(0)}(G_1) = \zeta_d G_d, F_{d_1}^{(0)}(G') = \zeta_d' G' \)
8. Find 0 \( \leq i_2, j_2 \) such that \( F_{d_1}^{(i_2)}(G_1) = F_{d_1}^{(j_2)}(G) \)
9. \( u_2 \leftarrow \sum_{i=0}^{i_2} f_i F_{d_1}^{(i)}(G_{d_1}) \mod d' \)
10. \( v_1 \leftarrow \sum_{i=0}^{i_2} f_i F_{d_1}^{(i)}(G) \mod d' \)
11. Output \( \alpha = \zeta^{k_1+k_2} d' \)

Random-walk Function \( F_s(G_0) \) that satisfied these requirements is defined as follows:

\[
F_s : \mathbb{G} \rightarrow \mathbb{Z}/s\mathbb{Z} \quad \text{(pseudo-random function, } s|\,(r-1)\text{),}
\]

\[
F_s : \mathbb{G} \rightarrow \mathbb{G},
\]

\[
F_s(G_0) : G_0 \rightarrow \zeta_s^{F_s(G_0)} G_0 \quad (s^i = \frac{r-1}{s}, \zeta_s = \zeta^s),
\]

\[
F_s^{(i)}(G_0) = \zeta_s^{F_s^{(i-1)}(G_0)} F_s^{(i-1)}(G_0). \quad \text{(1)}
\]

C. Distinguished Points Technique

In the kangaroo method, all the subsequent pairs of elements collide after the collision once happens. By using this characteristic, the number of stored data can be considerably reduced. This technique is called as the distinguished point technique \[17\].

When a chosen element has specified characteristic (e.g. the least significant 6 bits are all zero), the element is stored as a distinguished point. Even if firstly collided pair of points is not stored, collided pair of distinguished points are stored subsequently. Therefore, collided pair can be searched with reduced data.

By using this technique, the space complexity (also the number of elements) can be reduced to \( 1/w \) with arbitrary parameter \( w \). For example, the space complexity is \( O\left(\sqrt{(r-1)/d}/w\right) \) in step 1. On the other hand, the time complexity is increased to \( O\left(\log r\sqrt{(r-1)/d}+w\right) \) or \( O\left(\sqrt{(r-1)/d}+w\right) \) with KKM method. Since \( w \) is much smaller than \( \sqrt{(r-1)/d} \) in general, the increase of time complexity is negligible.

D. Cheon’s Algorithm with Kangaroo Method

With these techniques, the kangaroo method can be combined with Cheon’s algorithm (see Algorithm 2).
Algorithm 2': Improved Cheon's Algorithm with the kangaroo method

Input: \( G, G_1 = \alpha G, G_d = \alpha^d G \in \mathbb{G} \)
Output: \( \alpha \in \mathbb{Z}/r\mathbb{Z} \)

1. Find a generator \( \zeta \in (\mathbb{Z}/r\mathbb{Z})^* \)
2. \( \zeta_d \leftarrow \zeta^d, \zeta_d^\prime \leftarrow \zeta_d^d, d' \leftarrow (r - 1)/d \)
3. [Step 1] \( \sigma^{(0)}_{d,G} \leftarrow c_{1,1}, \sigma^{(0)}_{d',G} \leftarrow c_{1,2} \)
   \( (c_{1,1}, c_{1,2} \text{ are constant values}) \)
4. Find \( 0 \leq t_1, t_2 \) such that \( F_{d,1}^{(t_1)}(G_d) = F_{d,1}^{(t_2)}(G) \)
5. \( u_1 \leftarrow \sigma^{(1,1)}_{d,G}, v_1 \leftarrow \sigma^{(1,2)}_{d,G} \)
6. \( k_3 \leftarrow v_1 \mod d' \)
7. [Step 2] \( \sigma^{(0)}_{d,G} \leftarrow c_{2,1}, \sigma^{(0)}_{d',G} \leftarrow c_{2,2} \)
   \( (c_{2,1}, c_{2,2} \text{ are constant values}) \)
8. Find \( 0 \leq t_2, t_2 \) such that \( F_{d,2}^{(t_2)}(G_1) = \zeta^{k_1} F_{d,2}^{(t_2)}(G) \)
9. \( u_2 \leftarrow \sigma^{(2,1)}_{d,G}, v_2 \leftarrow \sigma^{(2,2)}_{d,G} \)
10. \( k_5 \leftarrow v_2 \mod d \)
11. Output \( \alpha = \zeta^{k_1 + k_2} \)

which is introduced and used for efficient pairing computation in practice [2]. Here are some parameters related to \( E:\)

- \( \sharp E(\mathbb{F}(3^{127})) = 25312879 \cdot 41757061638619 \cdot 399164334498301 \cdot r \) (202-bit),
- \( r = 9314856004962239626013099 \) (83-bit prime),
- \( r - 1 = 2 \cdot 3 \cdot 127 \cdot 2251 \cdot 5431 \cdot 7485427 \cdot 133582417 \),
- \( d = 2176458320181 \cdot 3 \cdot 5431 \cdot 133582417 \) (41-bit),
- \( \zeta = 12 \) (a minimum generator of \( (\mathbb{Z}/r\mathbb{Z})^* \)).

Here, \( \sharp E(\mathbb{F}(3^{127})) \) denotes the number of \( \mathbb{F}(3^{127}) \)-rational points on the elliptic curve \( E \), and \( d \) is chosen optimal so as to minimize the time complexity. Not that these parameters are same as those in [9], [10].

In the computation, an element \( z \in \mathbb{F}(3^{127}) = \mathbb{F}(3)[t]/(t^{127} + t^8 - 1) \) is represented as a polynomial \( z[126t^{126} + \cdots + z[1]t + z[0] \) \( [z[k] \in \{0, \pm 1\}) \), and, in addition, it is stored as a pair of 127-bit sequences

\[
\begin{align*}
z^+ &= (z^+[126], \ldots, z^+[1], z^+[0]) \\
z^- &= (z^-[126], \ldots, z^-[1], z^-[0])
\end{align*}
\]

where \( z^+[k] \) and \( z^-[k] \) are related to \( z[k] \): \( (z^+[k], z^-[k]) = (1, 0) \) if \( z[k] = 1, (z^+[k], z^-[k]) = (0, 1) \) if \( z[k] = -1 \) and \( (z^+[k], z^-[k]) = (0, 0) \) if \( z[k] = 0 \). A case \( (z^+[k], z^-[k]) = (1, 1) \) never occurs.

And, we generated a base-point \( G \) in the following manner same as [9], [10]. First, we generate a seed-point \( S \) such that all \( S.x^+[k] \) are 1 and all \( S.x^-[k] \) is 0, where \( S.x \) is the \( x \)-coordinate value of the point \( S \). The \( y \)-coordinate value \( S.y \) is obtained by \( S.x \) and the curve equation. Thus, we have

\[
\begin{align*}
S.x^+ &= 0x7F\text{FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF}
\quad S.x^- = 0x00000000000000000000000000000000
\quad S.y^+ = 0x5404929A80C20F08891241808610860
\quad S.y^- = 0x22C2008457299A0025628A60210EB71C.
\end{align*}
\]

Since the order of \( S \) equals to the order of the curve itself, a base-point \( G \) is obtained by \( G = \#E/rS \). Thus, we have

\[
\begin{align*}
G.x^+ &= 0x26150B200007930A412B0212208D8
\quad G.x^- = 0x081C3890DF7A004419643D502CD9401
\quad G.y^+ = 0x0BA754D97E065702986214815158B4
\quad G.y^- = 0x24500826020108020025065924EA2A0B.
\end{align*}
\]

Finally, target points \( G_1 = \alpha G \) and \( G_d = \alpha^d G \) are computed for \( \alpha = 3 \):

\[
\begin{align*}
G_1.x^+ &= 0x6C314812A8103C102C48AB02231B8A
\quad G_1.x^- = 0x0532B6E6A1C8CB010302268E21887B6
\quad G_1.y^+ = 0x2C01A00614346823EC6410210C05040,
\end{align*}
\]
\[ G_d.x^+ = 0x23154CC0519344D10C1D130C18200104, \]
\[ G_d.x^- = 0x506A00DDA6C802660800CB2A301B449, \]
\[ G_d.y^+ = 0x02809B34240C8282A0244A7401070C04, \]
\[ G_d.y^- = 0x356720C2585250784B43150ACDA0C301. \]

Here, \( \alpha = 3 \) is a solution we want to find by Cheon’s algorithm, which is as same as a value in [9], [10].

**B. Results**

In step 1 of Cheon’s algorithm with the kangaroo method, we establish two tables:

\[ T_{1.L} = \{ (\sigma_d(i), F_d^i(G_d)) \} \]

and

\[ T_{1.R} = \{ (\sigma_d(j), F_d^j(G)) \}. \]

Here, \( \sigma_d(i), \sigma_d(j) \) is 42-bit index values of chosen points. These values are calculated from previously chosen points.

In the same way of step 1, we establish two tables in step 2:

\[ T_{2.L} = \{ (\sigma_d(i), F_d^i(G_1)) \} \]

and

\[ T_{2.R} = \{ (\sigma_d(j), F_d^j(G)) \}. \]

In the Cheon’s algorithm with the kangaroo method, the total search space is too large to calculate all points. Thus, to establish two tables and to compare them, we have to be concurrently executed. Additionally, with the distinguished point technique, we stored only the points which the least significant 6-trit (ternary-digit) of \( x \)-axis value are all zero \( w = 3^6 \). For space saving purpose, each \( F_d^i(G_0) \) is digested as \( \text{LSB}_6(\text{MD}_5(F_d^i(G_0))) \), and these digested values are stored, which is same technique as [9].

In this instance, we parallelly established two tables with 2 core of 3GHz Core2Quad, and compared these tables with the other core. For 3 hours, about 1800 points (28 KByte) were stored in every two tables in step 1, and collided points are found. At this time, \( u_1 \leftarrow 1748512041946 \), \( v_1 \leftarrow 2439349585203 \), and \( k_1 \leftarrow 2439349585203 - 1748512041946 = 69083754257 \). Since \( \alpha^d = \zeta_k^d = 2116715807584984875228271 \), this experiment has solved step 1.

And for 2.75 hours, about 1600 points (26 KByte) were stored in every two tables in step 2. At this time, \( u_2 \leftarrow 1559364007743, v_2 \leftarrow 613947073386, \) and \( k_2 \leftarrow 613947073386 - 1559364007743 = 1231041385824(\text{mod} 2176458320181) \).

Finally,

\[ k \leftarrow k_1 + k_2 \cdot d' = 52686390264423397275434649, \]

and

\[ \alpha' \leftarrow \zeta_k^d (\text{mod} r) = 3. \]

Since \( \alpha' = \alpha \), this experiment has solved step 2. Therefore, it is confirmed that Cheon’s algorithm with the kangaroo method work correctly.

Note that, \( 1800 \times 3^6 \approx 2^{20.3} \) and \( 1600 \times 3^6 \approx 2^{20.2} \) points were computed per a table in practice respectively. And, in order to search \( i_1, j_1 \) that \( F_d^i(G_d) = F_d^{j_1}(G)G \) and \( i_2, j_2 \) that \( F_d^{i_2}(G_1) = F_d^{j_2}(G)G \), a naive method is used for two table comparison. Since tables are small, the time for table-comparison is negligible. Then, it is necessary to calculate \( F_d^i(G_0) \) from \( F_d^{i'}(G_0) \) for 8.2 msec on 1 core 3 GHz Core2Quad in this experiment.

**V. Comparison**

This section compares Cheon’s algorithm with the BSGS method and that with the kangaroo method.

**A. Previous Works**

In [9], [10], experimental results of Cheon’s algorithm with the BSGS method are shown. In the experiments, a pairing-friendly curve on \( \text{GF}(3^{127}) \) is used, which is also used in our experiment. This subsection briefly summarizes the results 2.

This experiment was conducted on 1 core of 3 GHz Core2Quad with straightforward implementation.

- In step 1, total 34 MByte table-making for 7 hours.
- In step 2, total 23 MByte table-making for 5 hours.

In the experiment, hash values of points are stored in these tables to reduce the total size of tables. For table-comparison, 1 hour in each step was required.

**B. Comparison**

Table I summarizes experimental results of Cheon’s algorithm with the BSGS method and the kangaroo method.

As mentioned above, the required space of Cheon’s algorithm with the kangaroo method is more efficient than that in the BSGS method. Additionally, Table I shows the required time of Cheon’s algorithm with the kangaroo method is also efficient. It seems that following three are causes of the fact.

- In [9] (BSGS method), all points are calculated.
- In our experiment, when collided points appeared in both tables, the calculation is stopped at once.

\[ 2 \text{In} [9], \text{they showed only an experimental result of step 1 with Cheon’s algorithm based with the BSGS method. And experiments of step 1 and step 2 were shown in} [10]. \text{Results in} [10] \text{is used in this paper.} \]
• In our experiment, collided points appeared according to the expectation.

Therefore, it is concluded that the time complexity of these two methods are almost same.

VI. SPEEDING-UP TECHNIQUE

In our experiment, we use no speeding-up techniques, but straightforward implementation. In this section, well-known speeding-up techniques are described.

As mentioned in section II-C, KKM method effectively reduces the time complexity to about \(1/\log r\). Excluding it, speeding-up techniques with using automorphism and parallel computing are known.

A. Using Automorphism

Some pairing-friendly curves use parameters which have automorphisms. If \(G\) has an automorphism, a random-walk function which chooses only representatives can be constituted, and the search space is reduced.

In this subsection, we explain the negation map and the Frobenius map as examples of automorphism.

1) Negation Map: When a minus point \(-G\) is efficiently computable from \(G\), the speeding-up technique with the negation map can be used in Cheon’s algorithm. The random-walk function which choose only plus points (or only minus points) reduces the time complexity to about \(1/\sqrt{2}\).

2) Frobenius Map: In the case of \(G = E(\text{GF}(p^m))\), Frobenius map \(\text{Fr}(x, y) \mapsto (x^p, y^p)\) on the group \(G\) is an automorphism of order \(m\). Here, \(E(\text{GF}(p^m))\) is an elliptic curve on \(\text{GF}(p)\).

When \(r - 1\) can be divided by \(m\), the speeding-up technique with Frobenius map can be used in Cheon’s algorithm. The random-walk function which choose only representatives reduce the time complexity to about \(1/\sqrt{m}\) (in ideal case of \(d \approx \sqrt{r}\)).

As mentioned above, using speeding-up techniques with automorphism, the time complexity can be reduced to about \(1/\sqrt{2m}\).

B. Parallel Computing

In order to parallelize Cheon’s algorithm, plural computers (cores) have to compute points in the group \(G\) independently.

As mentioned in III-E, constant value \(c_{1,1}, c_{1,2}\) in step 1 and \(c_{2,1}, c_{2,2}\) in step 2 are used in algorithm 2’ to change initial points for random walk function. Changing initial point, points calculated by random walk function constitute independent sequence. Therefore, algorithm 2’ can be parallelized by controlling these constant values. With parallel computing, the time complexity is virtually reduced to \(1/M\) with \(M\) computers.

Note: with parallel computing, two computers occasionally calculate collided point in a table \((T_{1,L, T_{1,R}, T_{2,L}}, \text{or } T_{2,R})\). At that time, the collision is detected and one of these computer is re-initialize with new constant value, as same as the case of self-collision.

VII. COST PREDICTION OF ANALYSIS AGAINST PAIRING-FRIENDLY CURVES

This section reports the cost prediction to analyze some pairing-friendly curves with Cheon’s algorithm.

In order to use experimental results mentioned above, we consider pairing-friendly curves on \(\text{GF}(3^{127})\) defined by:

\[
E_b : y^2 = x^3 - x + b, \quad b \in [-1, 1].
\]

A. Equation of Cost Computation

In order to cost computation of analysis with Cheon’s algorithm, we assume follows:

- Cost of a scalar multiplication on \(\text{GF}(3^{127})\) with straightforward implementation is 8.2 msec (see IV-B).
- Using KKM method, the cost is reduced \(1/\log r\).
- Cost of a scalar multiplication on \(\text{GF}(3^m)\) is \((m/127)^2\)-times that of on \(\text{GF}(3^{127})\).
- Cost of the algorithm based on kangaroo method is 2.8 times that of on BSGS-method (see III-D).
- Parameter \(d \approx \sqrt{r}\).
- Cost is reduced to \(1/\sqrt{2m}\) by automorphism technique.

Then, equation of cost computation is as follows:

\[
\text{Cost} = 4 \times \sqrt{r} \times 8.2/\log r \times (m/127)^2/\sqrt{2m} \text{ msec}.
\]

B. Cost Prediction

We show the targeted pairing-friendly curves and cost prediction of them.

1) \(m = 89, b = 1: [6]\)

\[
\#E(\text{GF}(3^{89})) = 7 \cdot 1069 \cdot 2137 \cdot 18193296720635112252099081035759771\quad (142\text{-bit}),
\]

\[
r = 181932967220635112252099081035759771\quad (118\text{-bit}),
\]

\[
r - 1 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 89 \cdot 8087 \cdot 4139171 \cdot 13010871 \cdot 2235089484239.
\]

\[
\text{Cost} = 4 \times \sqrt{r} \times 8.2/118 \times (89/127)^2/\sqrt{118} \text{ msec} \approx 8 \text{ hours}.
\]

2) \(m = 97, b = 1: [1], [2], [6], [12]\)

\[
\#E(\text{GF}(3^{97})) = 7 \cdot r \quad (154\text{-bit}),
\]

\[
r = 2726865189058261010774960798134976187171462721\quad (151\text{-bit}),
\]

\[
r - 1 = 26 \cdot 3 \cdot 5 \cdot 13 \cdot 17 \cdot 41 \cdot 43 \cdot 73 \cdot 97 \cdot 193 \cdot 577 \cdot 739 \cdot 769 \cdot 6481 \cdot 59632043 \cdot 42094721833.
\]

\[
\text{Cost} = 4 \times \sqrt{r} \times 8.2/151 \times (97/127)^2/\sqrt{154} \text{ msec} \approx 149 \text{ days}.
\]
The above-mentioned result is shown in Table II. These costs in this table are predicted with single-core CPU environment. As described in VI-B, the algorithm based on kangaroo method can be applied parallel computing. For example, if 100 cores of CPU are used, these cost is reduced to 1/100 virtually. Thus, if optimal \( d \) can be chosen, these pairing-friendly curves are analyzed by Cheon’s algorithm at reasonable cost.

VIII. CONCLUDING REMARKS

This paper has shown improving Cheon’s algorithm for experiments and reported cost prediction to analyze some pairing-friendly curves by Cheon’s algorithm. Our results showed that a portion of pairing-friendly curves can be analyzed by Cheon’s algorithm at reasonable cost if applicable data is obtained.

Cheon’s algorithm also solves problems based on some new problems such as \( \ell \)-WDH problem and \( \ell \)-SDH problem, on which some cryptographic schemes rely. Thus, the security evaluation from a viewpoint of Cheon’s algorithm is crucial for making the schemes secure in practice. In addition, selection of pairing-friendly curves also be important term to establish a real cryptographic system.

Table II

<table>
<thead>
<tr>
<th>( m )</th>
<th>( b )</th>
<th>( \mathbb{E} )</th>
<th>( r )</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>89</td>
<td>1</td>
<td>142-bit</td>
<td>118-bit</td>
<td>8 hours</td>
</tr>
<tr>
<td>97</td>
<td>1</td>
<td>154-bit</td>
<td>151-bit</td>
<td>149 days</td>
</tr>
<tr>
<td>103</td>
<td>1</td>
<td>164-bit</td>
<td>142-bit</td>
<td>22 days</td>
</tr>
<tr>
<td>127</td>
<td>-1</td>
<td>202-bit</td>
<td>83-bit</td>
<td>3 minutes</td>
</tr>
</tbody>
</table>

The cost prediction of Cheon’s algorithm against pairing-friendly curves is shown in Table II. These costs in this table are predicted with single-core CPU environment. As described in VI-B, the algorithm based on kangaroo method can be applied parallel computing. For example, if 100 cores of CPU are used, these cost is reduced to 1/100 virtually. Thus, if optimal \( d \) can be chosen, these pairing-friendly curves are analyzed by Cheon’s algorithm at reasonable cost.

3) \( m = 103, b = 1 \) \cite{[2]}

\[
\mathbb{E}(\mathbb{F}(3^{103})) = 7 \cdot 524683 \cdot r \ (164\text{-bit}),
\]
\[
r = 3788734765226760304517052348776005195050727 \ (142\text{-bit}),
\]
\[
r - 1 = 2 \cdot 3 \cdot 7 \cdot 17 \cdot 97 \cdot 103 \cdot 696239 \cdot 12353064397 \cdot 61752419775112892003,
\]
\[
\text{Cost} = 4 \times \sqrt{r} \times 8.2/142 \times (103/127)^2 / \sqrt{206} \text{ msec}
\]
\[
\simeq 22 \text{ days}.
\]

4) \( m = 127, b = -1 \) \cite{[2]}(same parameters in section IV)

\[
\mathbb{E}(\mathbb{F}(3^{247})) = 25312879 \cdot 41757061638619 \cdot 39916433449631 \cdot r \ (202\text{-bit}),
\]
\[
r = 9314856004986223962601399 \ (83\text{-bit}),
\]
\[
r - 1 = 2 \cdot 3 \cdot 127 \cdot 2251 \cdot 5431 \cdot 7485427 \cdot 133582417,
\]
\[
\text{Cost} = 4 \times \sqrt{r} \times 8.2/83 \times (127/127)^2 / \sqrt{254} \text{ msec}
\]
\[
\simeq 3 \text{ minutes}.
\]