Solving DLP with Auxiliary Input over an Elliptic Curve Used in TinyTate Library

Yumi Sakemi, Tetsuya Izu, Masahiko Takenaka, Masaya Yasuda

Solving DLP with Auxiliary Input over an Elliptic Curve Used in TinyTate Library

Yumi Sakemi[1], Tetsuya Izu[2], Masahiko Takenaka[2], and Masaya Yasuda[2]

[1] Okayama University
3-1-1, Tsushima-naka, Kita-ku, Okayama, 700-8530, Japan
[2] FUJITSU LABORATORIES Ltd.,
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki, 211-8588, Japan
{izu,takenaka,myasuda}@labs.fujitsu.com

Abstract. The discrete logarithm problem with auxiliary input (DLPwAI) is a problem to find α from $G,\ \alpha G,\ \alpha^d G$ in an additive cyclic group generated by G of prime order r and a positive integer d dividing $r-1$. The infeasibility of DLPwAI assures the security of some cryptographic schemes. In 2006, Cheon proposed a novel algorithm for solving DLPwAI. This paper shows our experimental results of Cheon's algorithm by implementing it with some speeding-up techniques. In fact, we succeeded to solve DLPwAI in a group with 128-bit order in 45 hours with a single PC on an elliptic curve defined over a prime finite field with 256-bit elements which is used in the TinyTate library.

1 Introduction

Let $\mathbb{G}$ be an additive cyclic group generated by G of prime order r. The discrete logarithm problem with auxiliary input (DLPwAI) is a problem to find α from $G,\ \alpha G,\ \alpha^d G \in \mathbb{G}$ and a positive integer d dividing $r-1$. The infeasibility of DLPwAI assures the security of some cryptographic schemes including Boneh-Boyen's ID-based encryption scheme [2] and Boneh-Gentry-Waters' broadcast encryption scheme [5]. In 2006, Cheon proposed a novel algorithm for solving DLPwAI [7, 8]. The time complexity of Cheon's algorithm (with KKM improvement) is $O\left(\sqrt{(r-1)/d}+\sqrt{d}\right)$, and especially when $d \approx \sqrt{r}$, the complexity becomes $O(\sqrt[4]{r})$, which is much efficient than that for solving DLP in general groups (which requires $O(\sqrt{r})$).

This paper implements Cheon's algorithm combined with the baby-step giant-step algorithm [16] as a sub-algorithm and some speeding-up techniques. Then, this paper reports experimental results of our implementation. In fact, we have successfully solved a DLPwAI in 45 hours with a single PC in a group with 128-bit order defined on an elliptic curve over a prime finite field with 256-bit elements, which is used in the TinyTate library [14] for implementing pairing cryptosystem in the embedded devices (see also Table 1. Note that Jao-Yoshida's result was not dedicated to an efficient implementation. Also note that Izu et al.'s result was implemented over a finite field with characteristics 3). Here, solving

Table 1. Required time for solving DLPwAI

	Size of r	Required Time
Jao, Yoshida [11]	60 bit	3 hours
Izu et al. [10]	83 bit	14 hours
This paper	**128 bit**	**45 hours**

DLP on the elliptic curve is regarded to be infeasible (since the order is 128-bit). As a feedback of our experimental results, it is better to avoid such elliptic curve when some cryptographic schemes are implemented. We also estimated the required time and memory for solving DLPwAI with larger r. According to our estimations, it would be difficult to solve DLPwAI with larger r by the same approach, namely, Cheon's algorithm combined with the baby-step giant-step algorithm.

2 Preliminaries

This section introduces the discrete logarithm problem with auxiliary input (DLPwAI) and Cheon's algorithm for solving DLPwAI [7, 8]. Implications of DLPwAI and Cheon's algorithm on cryptographic schemes are also explained.

2.1 Discrete Logarithm Problem with Auxiliary Input (DLPwAI)

Let $\mathbb{G} = \langle G \rangle$ be an additive group generated by G of prime order r. The discrete logarithm problem (DLP) in $\mathbb{G}$ is to find $\alpha \in \mathbb{Z}/r\mathbb{Z}$ on input $G,\ \alpha G \in \mathbb{G}$. In the general setting, the most efficient algorithms for solving DLP require $O(\sqrt{r})$ in time. In fact, Shanks' baby-step giant-step (BSGS) algorithm [16] requires $O(\sqrt{r})$ group operations in time and $O(\sqrt{r})$ group elements in space. On the other hand, Pollard's λ-algorithm also requires $O(\sqrt{r})$ in time, but much smaller elements in space.

In 2006, Cheon defined the discrete logarithm problem with auxiliary input (DLPwAI) as a variant of DLP [7, 8], where DLPwAI is a problem to find α on input $G,\ \alpha G,\ \alpha^d G$ and an integer d dividing $r - 1$ [3]. At the same time, Cheon proposed a novel algorithm for solving DLPwAI [7, 8]. Cheon's algorithm (together with Kozaki-Kutsuma-Matsuo's improvement [12]) requires $O\left(\sqrt{(r-1)/d} + \sqrt{d}\right)$ group operations in time. Especially, when $d \approx \sqrt{r}$, it only requires $O(\sqrt[4]{r})$ operations, which is much smaller than that required in the baby-step giant-step algorithm or in the λ-algorithm for solving DLP.

[3] There are many possible variations of DLPwAI. However, this paper only deals with this type.

Algorithm 1 Cheon's Algorithm

Input: : $G,\ G_1 = \alpha G,\ G_d = \alpha^d G \in \mathbb{G}$, $d \in \mathbb{Z}$ dividing $r-1$
Output: : $\alpha \in \mathbb{Z}/r\mathbb{Z}$

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

2.2 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 $\zeta \in (\mathbb{Z}/r\mathbb{Z})^*$ (Note that finding the generator ζ is easy). Here, such k is uniquely determined. To do so, Cheon's algorithm searches two integers $k_1,\ k_2$ such that $k = k_1 + k_2(r-1)/d$ satisfying $0 \le k_1 < (r-1)/d,\ 0 \le k_2 < d$ in the following two steps (see Algorithm 1).

Step 1 searches an integer k_1 such that $\alpha^d = \zeta_d^{k_1}$ for $\zeta_d = \zeta^d$, or equivalently, searches two integers $u_1,\ v_1$ such that

$$\alpha^d \zeta_d^{-u_1} = \zeta_d^{v_1 d_1}$$

satisfying $0 \le u_1, v_1 < \sqrt{(r-1)/d}$. Here, such $u_1,\ v_1$ are uniquely determined. In practice, Step 1 searches $u_1,\ v_1$ such that $\zeta_d^{-u_1} G_d = \zeta_d^{v_1 d_1} G$.

Then, Step 2 searches an integer k_2 such that $\alpha = \zeta^{k_1} \zeta_e^{k_2}$ for $\zeta_e = \zeta^{(r-1)/d}$ in the similar way, or equivalently, searches integers $u_2,\ v_2$ such that

$$\alpha \zeta^{-k_1} \zeta_e^{-u_2} = \zeta_e^{v_2 d_2}$$

satisfying $0 \le u_2, v_2 < \sqrt{d}$ (where $G_e = \zeta^{-k_1} G$). Here, such $u_2,\ v_2$ are uniquely determined. In practice, Step 2 searches $u_2,\ v_2$ such that $\zeta_e^{-u_2} G_1 = \zeta_e^{v_2 d_2} G$.

In Cheon's algorithm, searching u_1, v_1 in Step 1 and searching u_2, v_2 in Step 2 require another sub-algorithm. Since these problems are very similar to DLP in the general setting, the baby-step giant-step algorithm or the λ-algorithm can be used as a subroutine. Since this paper is interested in Cheon's algorithm combined with the baby-step giant-step algorithm only, the next section briefly describes its outline.

Baby-step Giant-step Algorithm The baby-step giant-step (BSGS) algorithm was introduced by Shanks in 1971 for solving DLP [16]. Instead of finding α directly, on input G and $G_1 = \alpha G$, BSGS searches two integers $i,\ j$ such

that $\alpha = i + jm$ and $0 \leq i,\ j < m = \lceil \sqrt{r} \rceil$. Here, such i, j are uniquely determined. Since $\alpha G = (i + jm)G = iG + jG'$ for $G' = mG$, we have a relation $G_1 - iG = jG'$.

BSGS 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 database. In the 2nd step (the giant-step), we compute

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

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

When BSGS algorithm is used in Step 1 of Cheon's algorithm, we establish two databases

$$\zeta_d^0 G_d,\ \zeta_d^{-1} G_d,\ \zeta_d^{-2} G_d,\ \ \ldots,\ \zeta_d^{-d_1} G_d$$

and

$$\zeta_d^0 G,\ \zeta_d^{d_1} G,\ \zeta_d^{2d_1},\ \ \ldots,\ \zeta_d^{d_1^2} G,$$

and searches a collision satisfying $\zeta_d^{-u_1} G_d = \zeta_d^{v_1 d_1} G$ among these databases. Thus, Step 1 in Cheon's algorithm combined with BSGS algorithm requires $2d_1 = 2\sqrt{(r-1)/d}$ elements in space. Similarly, Step 2 requires $2d_2$ elements in space.

KKM Improvement Cheon's algorithm requires the number of scalar multiplications for fixed elements G, G_1 and G_d. In 2007, Kozaki, Kutsuma and Matsuo introduced a precomputation table for such multiplications and reduced the time complexity of Cheon's algorithm from $O\left(\log r \left(\sqrt{(r-1)/d} + \sqrt{d}\right)\right)$ to $O\left(\sqrt{(r-1)/d} + \sqrt{d}\right)$.

Let us describe KKM improvement for a scalar multiplication γP ($\gamma \in \mathbb{Z}/r\mathbb{Z},\ P \in \mathbb{G}$) in the followings. For a fixed integer c (which will be optimized later) and $n = \lceil \sqrt[c]{r} \rceil$, obtain the n-array expansion of the scalar $\gamma = \sum_{i=0}^{c-1} \gamma_i n^i$ ($0 \leq \gamma_i < n$). For all $0 \leq i < c$ and $0 \leq j < n$, compute $S(i, j) = jn^i P$ and store them in a table in advance to the scalar multiplication. Then, the scalar multiplication γP is computed by the following way:

$$\begin{aligned} \gamma P &= \gamma_0 P + \gamma_1 nP + \cdots + \gamma_{c-1} n^{c-1} P \\ &= S(0, \gamma_0) + S(1, \gamma_1) + \cdots + S(c-1, \gamma_{c-1}). \end{aligned} \tag{1}$$

Since the precomputation table can be computed by at most c scalar multiplications and cn additions, KKM improvement reduces the time complexity of Cheon's algorithm by a factor of $\log r$.

Complexity of Cheon's Algorithm As a summary of this section, when Cheon's algorithm is combined with the baby-step giant-step algorithm and KKM improvement, the time complexity T and the space complexity S are evaluated by the followings:

$$T = O\left(\sqrt{(r-1)/d} + \sqrt{d}\right) \quad \text{(group operations)},$$
$$S = O\left(\max\left(\sqrt{(r-1)/d}, \sqrt{d}\right)\right) \quad \text{(group elements)}.$$

2.3 DLPwAI in Cryptographic Schemes

In recently proposed cryptographic schemes, new mathematical problems have been proposed and the infeasibility of such problems assure the security of the schemes. For example, ℓ-BDHE problem was introduced in Boneh-Gentry-Waters' broadcast encryption system [5]. Here, ℓ-BDHE problem is a problem to find $e(G, \hat{G})^{\alpha^{\ell+1}}$ for a given bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ on input $G,\ \alpha G, \ldots, \alpha^{\ell} G,\ \alpha^{\ell+2} G, \ldots,\ \alpha^{2\ell} G \in \mathbb{G}$ and $\hat{G} \in \hat{\mathbb{G}}$, where $\mathbb{G} = \langle G \rangle$, $\hat{\mathbb{G}} = \langle \hat{G} \rangle$, and $\mathbb{G}_T$ is a multiplicative group with order r. When $\ell > d$, Cheon's algorithm can be applied to the scheme: by finding α as DLPwAI, an answer of ℓ-BDHE problem is obtained. Thus, Cheon's algorithm is an attacking algorithm for such cryptographic schemes. Note that there are many other recently proposed problems to which Cheon's algorithm can be allied such as ℓ-SDH problem [3], ℓ-sSDH problem [4], ℓ-BDHI problem [2].

3 Implementation

This section describes our strategy for implementing Cheon's algorithm. We adopted the baby-step giant-step (BSGS) algorithm as a subroutine, and KKM improvement for the speeding-up.

3.1 BSGS Algorithm

Databases In step 1 of Cheon's algorithm, when BSGS algorithm is used, two databases

$$\mathrm{DB}_{1,\mathrm{B}} = \{\zeta_d^0 G_d,\ \zeta_d^{-1} G_d, \ldots,\ \zeta_d^{-i} G_d, \ldots, \zeta_d^{-d_1} G_d\}$$

and

$$\mathrm{DB}_{1,\mathrm{G}} = \{\zeta_d^0 G, \zeta_d^{d_1} G, \ldots,\ \zeta_d^{jd_1} G, \ldots,\ \zeta_d^{d_1^2} G\}$$

should be established. In our implementation, an element $\zeta_d^{-i} G_d$ in the database $\mathrm{DB}_{1,\mathrm{B}}$ is expressed by the following 12-byte representation

$$\underbrace{i}_{\text{4-byte}} \,||\, \underbrace{\mathrm{LSB}_{64}(\mathrm{MD5}(x(\zeta_d^{-i} G_d)||y(\zeta_d^{-i} G_d)))}_{\text{8-byte}}$$

where i is assumed to be 4-byte, $\mathrm{LSB}_{64}(\cdot)$ represents the least 64-bit (8-byte) of the data, MD5 is the hash function, and $x(G)$, $y(G)$ represent x- and y-coordinate values of a point G on an elliptic curve. In the following, i is identified as the index part and the rest is as the data part. Similarly, in the database $\mathrm{DB}_{1,\mathrm{G}}$, an element $\zeta_d^{jd_1}G$ is expressed by the following 12-byte representation

$$\underbrace{j}_{\text{4-byte}} \,||\, \underbrace{\mathrm{LSB}_{64}(\mathrm{MD5}(x(\zeta_d^{jd_1}G)||y(\zeta_d^{jd_1}G)))}_{\text{8-byte}}.$$

Thus, the databases $\mathrm{DB}_{1,\mathrm{B}}$ and $\mathrm{DB}_{1,\mathrm{G}}$ requires $12d_1$ bytes, respectively.

Database Search Then, in step 1 of Cheon's algorithm, it is needed to find two integers $u_1,\ v_1$ satisfying $\zeta_d^{-u_1}G_d = \zeta_d^{v_1 d_1}G$ among the databases $\mathrm{DB}_{1,\mathrm{B}}$ and $\mathrm{DB}_{1,\mathrm{G}}$. Since a straight-forward search (namely, compare all $\zeta_d^{-i}G_d$ in $\mathrm{DB}_{1,\mathrm{B}}$ and $\zeta_d^{jd_1}G$ in $\mathrm{DB}_{1,\mathrm{G}}$) is inefficient, we used the following bucket-sort like algorithm [1].

1. Divide each database into 64 sub-databases depending on the most significant 6-bit of the index part. When the most significant 6-bit of data $\zeta_d^{-i}G_d$ in $\mathrm{DB}_{1,\mathrm{B}}$ is ℓ $(0 \le \ell \le 63)$, it is stored in the sub-database $\mathrm{DB}_{1,\mathrm{B}}^{(\ell)}$. Similarly, when the most significant 6-bit of data $\zeta_d^{jd_1}G$ in $\mathrm{DB}_{1,\mathrm{G}}$ is ℓ $(0 \le \ell \le 63)$, it is stored in the sub-database $\mathrm{DB}_{1,\mathrm{G}}^{(\ell)}$.
2. Sort all sub-database $\mathrm{DB}_{1,\mathrm{G}}^{(\ell)}$ $(\ell = 0, \ldots, 63)$. by the comb-sort algorithm [6], which sorts N elements in $O(\log N)$.
3. For each ℓ, search a collision among $\mathrm{DB}_{1,\mathrm{B}}^{(\ell)}$ and $\mathrm{DB}_{1,\mathrm{G}}^{(\ell)}$. To do so, pick up an element $\zeta_d^{jd_1}G$ from the sub-database $\mathrm{DB}_{1,\mathrm{B}}^{(\ell)}$ and check whether the same element is in $\mathrm{DB}_{1,\mathrm{G}}^{(\ell)}$ by the binary search algorithm.

If a collision is found in step 3 for a certain ℓ, then, their indexes are what we required: set $u_1 \leftarrow i$ and $v_1 \leftarrow j$.

3.2 KKM Improvement

In our implementation, KKM improvement is also used for speeding-up Cheon's algorithm. Since our target group $\mathbb{G}$ is on an elliptic curve defined over a prime finite field with a mediate size, the affine coordinate system is used rather than the projective coordinate system. In the affine coordinate, every elliptic curve addition requires an inversion computation in the finite field. In order to avoid heavy operations, we used the Montgomery trick [13], which converts N inversion computations into 1 inversion and $3(N-1)$ multiplication computations. When the Montgomery trick is used in KKM improvement, only $O(\log_2 c)$ inversions are required.

4 Experimental Results

This section describes our experimental results of Cheon's algorithm for an elliptic curve used in the TinyTate library [14]. We successfully solved DLPwAI by our implementation in a group $\mathbb{G}$ with 128-bit order. Strongly note that DLP has been believed to be secure in the same group.

4.1 Parameters

We used an addition cyclic group $\mathbb{G} = \langle G \rangle$ with order r on an elliptic curve $y^2 = x^3 + x$ defined over a prime finite field $\mathbb{F}_p$ used in the TinyTate library [14] which was developed by Oliveria et al. for implementing the pairing-based cryptosystem in the embedded devices. Concrete values of these parameters are summarized in the following:

$$
\begin{aligned}
p &= \texttt{0x5387a1b6 93d85f28 8f131dd5 e7f9305c f4436019 a00f3181} \\
&\quad\ \texttt{168d7b20 8934d073} \ \ (256\text{-bit}) \\
&= 3778160688\ 9598235856\ 7455764726\ 5839472148\ 1625071533 \\
&\quad\ 3029839574\ 7614203820\ 7746163 \\
\#E &= \texttt{0x5387a1b6 93d85f28 8f131dd5 e7f9305c f4436019 a00f3181} \\
&\quad\ \texttt{168d7b20 8934d074} \ \ (256\text{-bit}) \\
&= 3778160688\ 9598235856\ 7455764726\ 5839472148\ 1625071533 \\
&\quad\ 3029839574\ 7614203820\ 7746164 \\
&= 2^2 \cdot 3^2 \cdot 1227703 \cdot 5024295160706223327986689772851 \cdot r \\
r &= \texttt{0x80000040 00000000 00000000 00000001} \ \ (128\text{-bit}) \\
&= 1701411885\ 3107163264\ 4604909702\ 696927233 \\
r-1 &= \texttt{0x80000040 00000000 00000000 00000000} \ \ (128\text{-bit}) \\
&= 2^{102} \cdot 3 \cdot 11 \cdot 251 \cdot 4051
\end{aligned}
$$

where $\#E$ denotes the number of points in $E(\mathbb{F}_p)$. In our implementation of Cheon's algorithm, we used the following parameters:

$$
\begin{aligned}
d &= 12682136550675316736 \ \ (64\text{-bit}) \\
&= 2^{60} \cdot 11 \\
\zeta &= 5 \\
\zeta_d &= \zeta^d = 1243133183\ 1021416944\ 7902414199\ 634645036 \\
d_1 &= \left\lceil \sqrt{(r-1)/d} \right\rceil = 3662760472 \ \ (32\text{-bit}) \\
d_2 &= \left\lceil \sqrt{d} \right\rceil = 3561198752 \ \ (32\text{-bit})
\end{aligned}
$$

Here, d is chosen to minimize the time complexity of Cheon's algorithm, and it is estimated that our implementation requires about $O(2^{32.75})$ group operations

Table 2. Computational Environment

CPU	Intel(R) Core^{TM} i7 2.93 GHz
OS	Ubuntu 10.04
Compiler	gcc 4.2.2
Library	GNU MP 4.1.2 [9]

and elements for solving DLPwAI. The generator ζ is chosen as the minimum generator of the multiplicative group $(\mathbb{Z}/r\mathbb{Z})^*$.

A base point G is randomly chosen from points in $E(\mathbb{F}_p)$ with order r. Then, coordinate values of G, $G_1 = \alpha G$, $G_d = \alpha^d G$ for our solution $\alpha = 3$ are as follows:

$$\begin{aligned}
x(G) &= 2120028877\ 3256318148\ 7254387784\ 2136477705\ 5392159948 \\
&\quad 4324389949\ 0272291266\ 930386 \\
y(G) &= 2676162370\ 5989368931\ 2040187896\ 9265522293\ 1214614323 \\
&\quad 9140635788\ 4068972949\ 5767328 \\
x(G_1) &= 1406565621\ 3797322149\ 8774526987\ 0546365700\ 1853001649 \\
&\quad 7338926577\ 6415100308\ 801614 \\
y(G_1) &= 3868330857\ 4106521926\ 0782358744\ 6121629591\ 2909889285 \\
&\quad 5061671768\ 3614580548\ 353865 \\
x(G_d) &= 3249689782\ 1175066681\ 3828703556\ 2385974940\ 1559994074 \\
&\quad 9555201487\ 6205365160\ 5880230 \\
y(G_d) &= 2017849900\ 8260892062\ 0757985589\ 8849092692\ 3717554232 \\
&\quad 0859082745\ 3474597173\ 7681072
\end{aligned}$$

4.2 Results

In the experiment, our implementation of Cheon's algorithm successfully found the solution $\alpha = 3$ in about 45 hours and 246 GByte by using a single PC (other environmental information are summarized in Table 2).

KKM Improvement In our implementation, the Montgomery trick is used for KKM improvement part. By experimental optimizations, we used parameters $n = 2^{16}$ and $c = 8$. The precomputation requires about 4 seconds, and each scalar multiplication requires about 27 μseconds. About 1.03 $m$seconds is required for a multiplication without KKM improvement, about 38 times speed-up was established. Also, about 54 μseconds is required with KKM improvement but without the Montgomery trick, about 2 times speed-up was established.

Step 1 Two databases $\mathrm{DB}_{1,\mathrm{B}}$ and $\mathrm{DB}_{1,\mathrm{G}}$ are generated in about 14.5 hours and 82 GByte memory with 4 parallel computations. Then, 1 hour is required

to divide these databases into 64 sub-databases $\mathrm{DB}_{1,\mathrm{B}}^{(\ell)}$ and $\mathrm{DB}_{1,\mathrm{G}}^{(\ell)}$ $(0 \leq \ell \leq 63)$. Since the parent databases $\mathrm{DB}_{1,\mathrm{B}}$ and $\mathrm{DB}_{1,\mathrm{G}}$, and their cache are stored in this divisions, $82 \times 3 = 246$ GByte are required.

Sorting a sub-database $\mathrm{DB}_{1,\mathrm{B}}^{(\ell)}$ required about 8 minutes, and searching a collision among $\mathrm{DB}_{1,\mathrm{B}}^{(\ell)}$ and $\mathrm{DB}_{1,\mathrm{G}}^{(\ell)}$ required about 2 minutes. In our experiment, we found a collision when $\ell = 39$ and obtained $u_1 = 2170110422$ and $v_1 = 846301393$. Thus, we found a partial solution

$$\begin{aligned} k_1 &= u_1 + v_1 d_1 \\ &= 3099799291849047918 \end{aligned}$$

in about 22.1 hours.

Step 2 Similarly to Step 1, the database generation required about 14.2 hours and 80 GByte. Dividing databases required about 1 hour and $80 \times 3 = 240$ GByte. Sorting a sub-database required about 7 minutes and searching a collision among sub-databases required about 2 minutes. We found a collision when $\ell = 52$ and obtained $u_2 = 1609744154$ and $v_2 = 718704617$. Thus, we found a partial solution

$$\begin{aligned} k_2 &= u_2 + v_2 d_2 \\ &= 2559449986726782138 \end{aligned}$$

in about 23.2 hours.

Consequently, we successfully obtained

$$\begin{aligned} k &= k_1 + k_2(r-1)/d \\ &= 343371056594041960083942329310843697 74 \end{aligned}$$

and the solution

$$\alpha = \zeta^k \bmod r = 3$$

in about 45.3 hours and 246 GByte memory.

4.3 Estimations

Based on our experimental results described in the previous section, we estimate required time (in the worst case) and memory for solving DLPwAI with larger r. Here, we do not consider the parallel processing. We assume that the parameter d can be chosen as large as $\sqrt{r}$, namely, required time in Step 1 and Step 2 are almost same.

Let T_{C} be the required time for generating databases. Since a scalar multiplication is computed in 27 μseconds, T_{C} can be evaluated by

$$T_{\mathrm{C}} = 2 \times 27 \times 10^{-6} \times 2\sqrt[4]{r} \quad [\text{Seconds}]. \tag{2}$$

Table 3. Cost Estimations

r	T_{C}	T_{S}	$T = T_{\mathrm{C}} + T_{\mathrm{S}}$	S_{C}
128 bit	6 Days	17 Hours	7 Days	288 GByte
132 bit	11 Days	35 Hours	13 Days	576 GByte
136 bit	22 Days	74 Hours	25 Days	1152 GByte
140 bit	45 Days	6 Days	51 Days	2304 GByte

Then, let us evaluate the required time T_{S} for the database search. Since the sorting are dominant procedures, we neglect time for other parts. In our implementation, the comb-sort algorithm requires $O(\log N)$ for sorting N elements, and a sort in a sub-database requires 8 minutes for 128-bit r, T_{S} can be evaluated by

$$T_{\mathrm{S}} = 2 \times 64 \times (8 \times 60) \times \frac{(\sqrt[4]{r}) \log (\sqrt[4]{r}/64)}{(2^{32}) \log (2^{26})} \quad [\text{Seconds}].$$

On the other hand, the required memory S_{C} can be evaluated by

$$S_{\mathrm{C}} = 12 \times \sqrt[4]{r} \times 3 \times 2 \ [\text{Bytes}].$$

By using these evaluations, estimated required time and memory for various sizes of r are summarized in Table 3.

According to Table 3, solving DLPwAI seems to be feasible even if r is 140-bit since the required time is about 50 days. However, the required memory is beyond 2 TByte in this case. In the computational environment we used, and in most environments, dealing with such huge memory is too difficult to proceed. Thus, it is concluded that solving DLPwAI with 140-bit r is infeasible by Cheon's algorithm combined with BSGS algorithm. In order to solve such larger problems, Cheon's algorithm combined with the λ-algorithm or the kangaroo algorithm would be employed.

Next, let us compare our results to the previous experiments (summarized in Table 1) when r is 128-bit by the extrapolation. According to the time complexity of Cheon's algorithm, Jao and Yoshida's implementation [4] would require $3 \times \sqrt{2^{64}/2^{30}} = 393216$ hours (16384 days) and Izu et al.'s implementation would require $14 \times \sqrt{2^{64}/2^{42}} = 28672$ hours (1195 days) (Table 4). Even if the parallel computation is applied, solving DLPwAI is infeasible with 128-bit r by these implementations.

5 Concluding Remarks

This paper succeeded to solve a discrete logarithm problem with auxiliary input (DLPwAI) in 45 hours with a single PC in a group with 128-bit order defined on

[4] Note that Jao-Yoshida's result was not dedicated to an efficient implementation.

Table 4. Estimated Time for 128-bit r

	Estimated Time
Jao, Yoshida [11]	16384 Days
Izu et al. [10]	1195 Days
This paper	7 Days

an elliptic curve over a prime finite field with 256-bit elements, which are used in TinyTate library for implementing pairing cryptosystem in the embedded devices. If cryptographic schemes based on problems such as ℓ-BDE problem, ℓ-SDH problem, ℓ-sSDH problem or ℓ-BDHI problem are implemented, TinyTate library should avoid using such weak parameters. However, there are pairing-based cryptographic schemes which are not effected by Cheon's algorithm such as Boneh-Franklin's ID-based encryption scheme.

Acknowledgements

The authors would like to thank Yoshitaka Morikawa and Yasuyuki Nogami in Okayama University for their supports on this research.

References

1. K. Aoki and H. ,Ueda, "Sieving Using Bucket Sort", *ASIACRYPT 2004*, LNCS 3329, pp. 92-102, Springer, 2004.
2. D. Boneh and X. Boyen, "Efficient Selective-ID Secure Identity-based Encryption without Random Oracles", *EUROCRYPT 2004*, LNCS 3027, pp. 223-238, Springer, 2004.
3. D. Boneh and X. Boyen, "Short Signatures without Random Oracles", *EUROCRYPT 2004*, LNCS 3027, pp. 56-73, Springer, 2004.
4. D. Boneh, X. Boyen, and E. Goh, "Hierarchial Identity Based Encryption with Constant Size Ciphertext", *EUROCRYPT 2005*, LNCS 3494, pp. 440-456, Springer, 2005.
5. D. Boneh, C. Gentry, and B. Waters, "Collusion Resistant Broadcast Encryption with Short Ciphertext and Private Keys", *CRYPTO 2005*, LNCS 3621, pp. 258-275, Springer, 2005.
6. R. Box et al., "A Fast Easy Sort", *Computer Journal of Byte Magazine*, vol. 16, no. 4, pp.315-320, 1991.
7. J. H. Cheon, "Security Analysis of Strong Diffie-Hellman Problem", *EUROCRYPT 2006*, LNCS 4004, pp. 1-11, Springer, 2006.
8. J.H. Cheon, "Discrete Logarithm Problems with Auxiliary Inputs", *Journal of Cryptology*, vol. 23, no. 3, pp.457-476, Springer, 2010.
9. GNU MP, `http://gmplib.org/`
10. T. Izu, M. Takenaka, and M. Yasuda, "Experimental Results on Cheon's Algorithm", *WAIS 2010*, pp. 625-630 in the proceedings of *ARES 2010*, IEEE Computer Science, 2010.

11. D. Jao and K Yoshida, "Boneh-Boyen Signatures and the Strong Diffie-Hellman Problem", *Pairing 2009*, LNCS 5671, pp. 1-16, Springer, 2009.
12. S. Kozaki, T. Kutsuma, and K. Matsuo, "Remarks on Cheon's Algorithms for Pairing-related Problems", *Pairing 2007*, LNCS 4575, pp. 302-316, Springer, 2007.
13. P. Montgomery, "Speeding the Pollard and Elliptic Curve Methods of Factorization", *Math. Comp.*, vol. 48, no. 177, pp. 243-264, 1987.
14. L. Oliveira, J. López, and R. Dahab, "TinyTate: Identity-Based Encryption for Sensor Networks", IACR Cryptology ePrint Archive, Report 2007/020, 2007.
15. J. Pollard, "Monte Carlo Methods for Index Computation (mod p)", *Math. Comp.*, vol. 32, pp. 918-924, 1978.
16. D. Shanks, "Class Number, a Theory of Factorization, and Genera", *Proc. of Symp. Math. Soc.*, vol. 20, pp. 41-440, 1971.