Memoirs of the Faculty of Engineering, Okayama University, Vol. 47, pp. 25-32, January 2013

Lazy Random Walk Efficient for Pollard's Rho Method Attacking on G3 over Barreto-Naehrig Curve (Corrected)

Kenta NEKADO*, Yusuke TAKAI*, and Yasuyuki NOGAMI*
Graduate School of Natural Science and Technology, Okayama University
3–1–1, Tsushima–naka, Kita–ward, Okayama–city, Okayama–pref., 700–8530, Japan

(Received December 11, 2012)

Pairing–based cryptosystems are well implemented with Ate–type pairing over Barreto–Naehrig (BN) curve. Then, for instance, their securities depend on the difficulty of Discrete Logarithm Problem (DLP) on the so–denoted $\mathbb{G}_3$ over BN curve. This paper, in order to faster solve the DLP, first proposes to utilize Gauss period Normal Basis (GNB) for Pollard's rho method, and then considers to accelerate the solving by an adoption of *lazy random walk*, namely *tag tracing* technique proposed by Cheon et al.

1 Introduction

After year 2000, *pairing*–based cryptosystems have been contrived such as ID–based encryption [1, 2], group signature [3], and timed–release encryption [4]. Pairing is defined as a bilinear map from an additive group or 2 additive groups over elliptic curve to a multiplicative group in finite field. The securities of pairing–based cryptosystems depend on the difficulties of *Discrete Logarithm Problem* (DLP) on a certain multiplicative group in finite field, and *Elliptic Curve DLP* (ECDLP) on (a) certain additive group(s) over elliptic curve. Several kinds of elliptic curves to make these difficulties optimal have been proposed, and are called *pairing–friendly curves* [5]. Among them, Barreto–Naehrig (BN) curve [6], which this paper focuses on, is known as one of the elliptic curves which will provide the optimal difficulties of both DLP and ECDLP after year 2030 [7], that is, which will reasonably achieve the 128–bit security [5].

As fast pairings over BN curve, the improved Ate pairings [8] such as R–ate [9], optimal Ate [10], and Xate [11] pairings have been contrived. These Ate–type pairings are bilinear maps from the Cartesian product of the so–denoted $\mathbb{G}_1$ and $\mathbb{G}_2$, which are 2 certain cyclic additive subgroups over elliptic curve, to the so–denoted $\mathbb{G}_3$, which is a certain cyclic multiplicative subgroup in finite field. The securities of the cryptosystems based on Ate–type pairing depend on the difficulties of both the ECDLPs on $\mathbb{G}_1$ and $\mathbb{G}_2$, and the DLP on $\mathbb{G}_3$. Thus, for any adoption of Ate–type pairing, it is one of the most important to explore the possibilities of solving these ECDLPs and DLP.

As approaches to solve DLP and ECDLP, for example, most researchers adopt Pollard's rho method [12] and its varieties. Let a group $\mathbb{G}$ be an attack target for solving DLP or ECDLP. This rho method needs the procedure called *random walk*, which iterates to randomly generate elements in $\mathbb{G}$ until a newly generated element in $\mathbb{G}$ corresponds to one of the previously generated elements. The corresponding element is called the *self–collision element.* The number of the iterations required to solve a DLP or ECDLP is stochastically given [13], and it can be reduced by applying *automorphism technique* [14]. Thus, the authors have already demonstrated a possibility to solve the ECDLP on $\mathbb{G}_2$ with the rho method applied the automorphism technique, namely the so–called *skew–Frobenius map* [15]. This paper explores a possibility to solve the DLP on $\mathbb{G}_3$ with the rho method.

Since $\mathbb{G}_3$ over BN curve is the multiplicative subgroup in a certain extension filed, the random walk mainly consists of multiplication in the extension field. Additionally, as an automorphism available for the rho method, the extension field has the so–called *Frnobenius map.* Thus, in the case of using the rho method, the calculation times of both multiplication and Frobenius mapping become very important. Note that the solvers need not to use the extension field adopted as $\mathbb{G}_3$ in the actual cryptosystems. By applying *basis conversion* technique [16], they can select an extension field convenient for solving the DLP on $\mathbb{G}_3$. Thus, this paper focuses on the extension field constructed by *Gauss period Normal Basis* (GNB) [17] which achieves fast multiplication and needs no arithmetic operations for Frobenius mapping. And, this paper first clarifies the computation

*E-mail: {nekado, takai, nogami}@trans.cne.okayama–u.ac.jp

times of solving a DLP on $\mathbb{G}_3$ when adopting the extension field.

On the other hand, as an acceleration technique of the rho method to solve DLP on finite fields, Cheon et al. have proposed *tag tracing technique* [18]. This technique allows the random walk to skip several steps by predicting the originally generated elements. In other words, the random walk seems to be *lazy.* This paper optimizes this technique for solving the DLP on $\mathbb{G}_3$, and shows that the optimized technique is more superior than the automorphism technique.

Notation : $\mathbb{Z}_r$ denotes a finite integer ring with an order r, $\mathbb{F}_p$ denotes a prime field with a characteristic p, $\mathbb{F}_{p^m}$ denotes an m–th degree extension field over $\mathbb{F}_p$ $\mathbb{F}^*_{p^m}$ denotes the multiplicative group in $\mathbb{F}_{p^m}$, and $E(\mathbb{F}_{p^m})$ denotes an elliptic curve additive group over $\mathbb{F}_{p^m}$. For 2 positive integers m and n, $m \mid n$ means that m divides n, $m \nmid n$ means that m does not divide n, and $m \parallel n$ means that m divides n only once.

2 Attack Target and Method

This paper tries to attack the output of Ate–type pairing over Barreto–Naehrig (BN) curve, namely the so–denoted $\mathbb{G}_3$. Concretely, the *Discrete Logarithm Problem* (DLP) on $\mathbb{G}_3$ aspires to be solved with Pollard's rho method. Therefore, this section introduces the attack target $\mathbb{G}_3$, Pollard's rho method, and its acceleration technique.

2.1 $\mathbb{G}_3$ of Ate–type Pairing over BN Curve

Ate–type pairing [8, 9, 10, 11] is a class of bilinear and negligible map. Let it be given by the following α,

$$\alpha : \mathbb{G}_2 \times \mathbb{G}_1 \to \mathbb{G}_3 = \mathbb{F}^*_{p^k}/(\mathbb{F}^*_{p^k})^r, \tag{1a}$$

$$\mathbb{G}_1 = \{E(\mathbb{F}_{p^k})/[r]E(\mathbb{F}_{p^k})\} \cap \mathrm{Ker}(\phi - [1]), \tag{1b}$$

$$\mathbb{G}_2 = \{E(\mathbb{F}_{p^k})/[r]E(\mathbb{F}_{p^k})\} \cap \mathrm{Ker}(\phi - [p]), \tag{1c}$$

where "Ker" and ϕ respectively denote the *kernel* of a homomorphism and the *Frobenius map.* Additionally, the above k is the smallest positive integer such that $r|(p^k - 1)$, and is called *embedding degree.*

BN curve [6] is a class of ordinary pairing–friendly curves with embedding degree $k = 12$. In the case of BN curve, the p and r of Eq. (1) are given by

$$p(\chi) = 36\chi^4 + 36\chi^3 + 24\chi^2 + 6\chi + 1, \tag{2a}$$

$$r(\chi) = 36\chi^4 + 36\chi^3 + 18\chi^2 + 6\chi + 1, \tag{2b}$$

where χ is a positive or negative integer such that $p(\chi)$ becomes a prime number. In order to achieve the 128–bit security for Ate–type pairing over BN curve, the p and r of Eq. (2) must be assigned 256–bit prime numbers.

As the above, the attack target $\mathbb{G}_3$ is a certain cyclic subgroup with the order r in $\mathbb{F}_{p^{k(=12)}}$.

2.2 Pollard's Rho Method with Automorphism

This paper considers how to solve an DLP on $\mathbb{G}_3$ over BN curve with Pollard's rho method, that is, how to derive a exponent c in $\mathbb{Z}_r$ from X and $Y(= X^c)$ in $\mathbb{G}_3$ by using **Alg.** 1. The procedure of iterations from **Step** 8 to 11 is the *random walk.* The number of the iterations is given by average $\sqrt{\pi r/2}$ [13].

For this rho method, the acceleration technique with equivalence classes derived by an automorphism [14] can be applied. In the case of $\mathbb{G}_3$ over BN curve, for any element W in $\mathbb{G}_3$, there exists the *Frobenius map* $W \in \mathbb{G}_3 \mapsto W^p \in \mathbb{G}_3$ as an efficient automorphism, and the conjugate set $\{W, W^p, W^{p^2}, \ldots, W^{p^{k-1=11}}\}$ is considered as an equivalence class. Thus, the rho method can be improved by canonically deciding a unique representative for each equivalence class. The improvement is achieved by adopting **Alg.** 2 to **Alg.** 1. Then, the number of the iterations can be theoretically reduced to $\sqrt{\pi r/24}$ [14]; however, every representative decision shown in **Alg.** 2 must be efficiently performed. Additionally, this automorphism technique occurs that the random walk gets trapped into *fruitless cycles.* Thus, in order to break away from the trap, for example, the countermeasure as introduced in [20] must adapt.

Algorithm 1: Solving with Pollard's rho method [12, 19]

```
Input: $X, Y\ (= X^c) \in \mathbb{G}_3$.
Output: $c \in \mathbb{Z}_r$.
1  for $i = 0$ to $T - 1$ do
2      assign random elements in $\mathbb{Z}_r$ to $a_i$, $b_i$.
3      $W_i \leftarrow X^{a_i} \cdot Y^{b_i}$.
4  assign random elements in $\mathbb{Z}_r$ to $a_T$, $b_T$.
5  $W_T \leftarrow X^{a_T} \cdot Y^{b_T}$.
6  for $i = T + 1$ to $r - 1$ do
7      $f \leftarrow \eta(W_{i-1})$.
8      $a_i \leftarrow a_{i-1} + a_f$, $b_i \leftarrow b_{i-1} + b_f$, $W_i \leftarrow W_{i-1} \cdot W_f$.
11     if $W_i = W_j\ (0 \le j < i)$ then go to Step 12.
12 $c \leftarrow -(a_i - a_j)/(b_i - b_j)$.
```

† η denotes a *hash function.* For example, let w denote the 1–st element of $W \in \mathbb{G}_3$, then it is given by $\eta(W) = w \bmod n$.

Algorithm 2: Representative decision

```
9  decide the representative $W_i^{p^j}$ from the
       equivalence class $\{W_i, W_i^p, W_i^{p^2}, \ldots, W_i^{p^{k-1=11}}\}$.
10 $a_i \leftarrow p^j a_i$, $b_i \leftarrow p^j b_i$, $W_i \leftarrow W_i^{p^j}$.
```

3 Extension Field Efficient for Attack

As introduced in **Sec.** 2.2, in order to more efficiently perform the rho method with the Frobenius map, the Fronibeus mapping in $\mathbb{G}_3$ must be fast carried out. Thus, as how to construct an extension field such that the Frobenius mapping requires no arithmetic operations, this section introduces *Gauss period Normal Basis* (GNB). Additionally, as shown in **Alg.** 1, the main operation in every random walk is a multiplication in $\mathbb{G}_3$. Thus, this section also introduces some fast multiplication algorithms.

3.1 Extension Field Constructed by GNB

GNB [17] requires not only a characteristic p and an extension degree m but also a positive integer h which satisfies the following conditions.

Condition 1 (The parameter h of GNB)

1. $n = hm+1$ is a prime number not equal to p.
2. $\gcd(hm/e, m) = 1$, where e is the order of p in $\mathbb{F}_n$.

Here, let d and β respectively denote any primitive h–th power root of unity in $\mathbb{F}_n$ and any primitive n–th power root of unity in $\mathbb{F}_{p^e}$, and then GNB is defined as follows.

Definition 1 The conjugate set provided by the following γ forms normal basis in $\mathbb{F}_{p^m}$.

$$\{\gamma, \gamma^p, \cdots, \gamma^{p^{m-1}}\}, \quad \gamma = \sum_{l=0}^{h-1} \beta^{d^l} \tag{3}$$

This paper calls this normal basis type–$\langle h, m \rangle$ GNB. □

Because type–$\langle h, m \rangle$ GNB can be prepared whenever $4p \nmid m(p-1)$ [17], it is available in $\mathbb{F}_{p^m}$ for every pair of characteristic p and extension degree m when $p > m$.

Since type–$\langle h, m \rangle$ GNB is normal basis, a Frobenius mapping in the extension field constructed by this basis requires no arithmetic operations. Concretely, a Frobenius mapping can be performed with only a cyclic shift as follows.

$$\begin{aligned} C = \sum_{i=0}^{m-1} c_i \gamma^{p^i} \in \mathbb{F}_{p^m}, \quad c_i \in \mathbb{F}_p \\ \mapsto \ C^p = \sum_{i=0}^{m-1} c_{(i-1) \bmod m} \gamma^{p^i} \in \mathbb{F}_{p^m}. \end{aligned} \tag{4}$$

On the other hand, generally, a multiplication in the extension field constructed by type–$\langle h, m \rangle$ GNB requires more additions as the parameter h becomes larger. Thus, in order to more efficiently perform the multiplication, the parameter h should be as small as possible. Moreover, it is the most desirable that $h = 1$ or $h = 2$ because the multiplication in the extension field constructed by type–$\langle h=1, m \rangle$ or type–$\langle h=2, m \rangle$ GNB are the most efficient. Actually, type–$\langle h=1, m \rangle$ and type–$\langle h=2, m \rangle$ GNBs are respectively called type–I and type–II Optimal Normal Bases (ONBs).

When type–$\langle h, m \rangle$ GNB is applied to construct the $\mathbb{G}_3$ convenient for solving the DLP, the following extension fields can be considered.

1. The $\mathbb{F}_{p^{12}}$ constructed by type–$\langle h_{12}, m=12 \rangle$ GNB
2. The $\mathbb{F}_{(p^3)^4}$ towering with type–$\langle h_4, m=4 \rangle$ over $\mathbb{F}_{p^3}$ constructed by type–$\langle h_3, m=3 \rangle$ GNB
3. The $\mathbb{F}_{(p^4)^3}$ towering with type–$\langle h_3, m=3 \rangle$ over $\mathbb{F}_{p^4}$ constructed by type–$\langle h_4, m=4 \rangle$ GNB

Below, this section introduces some multiplication algorithms available for these extension fields, and discusses the calculation costs of a multiplication.

3.2 Lazy Reduction

Before some multiplication algorithm in extension field are introduced, this subsection introduces *lazy reduction* technique [21, 22] which is an acceleration technique of arithmetic operations in extension field .

A multiplication in $\mathbb{F}_{p^m}$ requires some multiplication in $\mathbb{F}_p$, and thus generally requires some reductions modulo p as many as the multiplications in $\mathbb{F}_p$. The calculation cost of a reduction modulo p is not small but rather very heavy. In the case of lazy reduction technique, the every multiplication in $\mathbb{F}_p$ necessary for a multiplication in $\mathbb{F}_{p^m}$ is repraced an integer multiplication, and reductions modulo p are performed only in the last of the multiplication in $\mathbb{F}_{p^m}$. Then, the originally necessary additions in $\mathbb{F}_p$ are replaced to integer addition with *double precision*, where *double precision* means $(\lceil \log_2 p \rceil \times 2)$–bit. By applying lazy reduction technique in the same way as Aranha et al.'s implementation [22], the number of reductions modulo p which are necessary for a multiplication in $\mathbb{F}_{p^{12}}$, $\mathbb{F}_{(p^3)^4}$ or $\mathbb{F}_{(p^4)^3}$ can be reduced to 12. As the above, a multiplication in $\mathbb{F}_{p^m}$ becomes to require some integer addition with double precision; however, it can be accelerated since many reductions modulo p can be deleted. This paper supposes that lazy reduction technique is applied for multiplication algorithms in extension field.

Below, $\tilde{M}_1$, A_1, $\tilde{A}_1$, and R_1 respectively denote the calculation costs of an integer multiplication, an addition in $\mathbb{F}_p$ (an integer addition with single precision), an integer addition with double precision, a reduction modulo p. Note that this paper supposes that the calculation costs of a subtraction in $\mathbb{F}_p$ and an integer subtraction with double precision are respectively given by A_1 and $\tilde{A}_1$. In this paper, the calculation costs are given by the above notations.

3.3 Cyclic Vector Multiplication Algorithm

As an efficient multiplication algorithm in the extension field constructed by type–$\langle h, m \rangle$ GNB, the authors have proposed type–$\langle h, m \rangle$ Cyclic Vector Multiplication Algorithm (CVMA) [23]. Type–$\langle h, m \rangle$ CVMA is illustrated in **Alg.** 3. With type–$\langle h, m \rangle$ CVMA, the calculation cost of a multiplication M_m is given by

$$\begin{aligned} M_m = &\{m(m+1)/2\}M_1 + \{m(m-1)\}A_1 \\ &+ \begin{cases} \{(hm+2)(m-1)/2\}\tilde{A}_1 + \tilde{H}_1 & (h\text{: odd}) \\ \{hm(m-1)/2\}\tilde{A}_1 & (h\text{: even}) \end{cases}, \end{aligned} \tag{5}$$

where $\tilde{H}_1$ denotes the calculation cost of an integer multiplication by the integer h with double precision.

Algorithm 3: Type–$\langle h, m\rangle$ CVMA [23]

```
Input: $X = \sum_{i=0}^{m-1} x_i\gamma^{p^i}$, $Y = \sum_{i=0}^{m-1} y_i\gamma^{p^i}$, $x_i, y_i \in \mathbb{F}_p$.
Output: $Z = X \cdot Y = \sum_{i=0}^{m-1} z_i\gamma^{p^i}$, $z_i \in \mathbb{F}_p$.
Precomputation steps:
get an $h$–th power root $d$ of unity in $\mathbb{F}_n$.
$\epsilon[0] \leftarrow m$.
for $i = 0$ to $m-1$ do
    for $l = 0$ to $h-1$ do $\epsilon\left[\langle\!\langle p^i d^l \rangle\!\rangle\right] \leftarrow i$.
for $i = 0$ to $m-2$ do
    for $j = i+1$ to $m-1$ do
        for $l = 0$ to $h-1$ do
            $\lambda[i,j,l] \leftarrow \epsilon\left[\langle\!\langle p^i + p^j d^l \rangle\!\rangle\right]$.
Main calculation steps:
for $i = 0$ to $m$ do $v_i \leftarrow 0$.
for $i = 0$ to $m-2$ do
    for $j = i+1$ to $m-1$ do
        $u \leftarrow (x_i - x_j)(y_i - y_j)$.
        for $l = 0$ to $h-1$ do
            $v_{\lambda[i,j,l]} \leftarrow v_{\lambda[i,j,l]} + u$.
if $h$ is odd then
    $w \leftarrow hv_m$.
    for $i = 0$ to $m-1$ do $z_i \leftarrow -x_iy_i - v_i + w$.
else  for $i = 0$ to $m-1$ do $z_i \leftarrow -x_iy_i - v_i$.
```

3.4 Karatsuba's Multiplication Algorithm

Since the γ of type–$\langle h=1, m\rangle$ GNB shown in Eq. (3) satisfy that $\gamma^{m+1} = 1$, this basis forms not only normal basis but also *pseudo* plynomial basis as follows.

$$\{\gamma, \gamma^p, \cdots, \gamma^{p^{m-1}}\} = \{\gamma, \gamma^2, \cdots, \gamma^m\}. \qquad (6)$$

Thus, Karatsuba's multiplication algorithm [25] can be applied for the extension field constructed by type–$\langle h=1,\ m\rangle$ GNB. Below, this subsection shows Karatsuba's multiplication in the $\mathbb{F}_{p^{12}}$ constructed by type–$\langle h_{12}=1,\ m=12\rangle$ GNB, and that in the $\mathbb{F}_{p^4}$ constructed by type–$\langle h_4=1,\ m=4\rangle$ GNB. On the other hand, according to **Cond**. 1, this subsection does not discuss about the $\mathbb{F}_{p^3}$ since there exists no type–$\langle h_3=1, m=3\rangle$ GNB.

The case of the $\mathbb{F}_{p^{12}}$: Let 2 elements X and Y in $\mathbb{F}_{p^{12}}$ be represented by pseudo polynomial basis as

$$X = (x_0 + x_1\gamma^6)\gamma, \qquad Y = (y_0 + y_1\gamma^6)\gamma, \qquad (7a)$$

$$x_i = x_{i,0} + x_{i,1}\gamma^3, \qquad y_i = y_{i,0} + y_{i,1}\gamma^3, \qquad (7b)$$

$$x_{i,j} = x_{i,j,0} + x_{i,j,1}\gamma + x_{i,j,2}\gamma^2, \quad x_{i,j,l} \in \mathbb{F}_p,$$
$$y_{i,j} = y_{i,j,0} + y_{i,j,1}\gamma + y_{i,j,2}\gamma^2, \quad y_{i,j,l} \in \mathbb{F}_p. \qquad (7c)$$

Then, the polynomial product $Z = X \cdot Y$ is obtained as

$$Z = u_0\gamma^2 + u_1\gamma^8 + u_2\gamma^{14}, \quad u_0 = x_0 \cdot y_0, \quad u_2 = x_1 \cdot y_1,$$
$$u_1 = (x_0 - x_1)(y_1 - y_0) + u_0 + u_2. \qquad (8)$$

According to Eq. (8), the polynomial product requires 3 multiplications of sextic polynomial whose coefficients are single precision integers, 2 additions of sextic polynomial whose coefficients are elements in $\mathbb{F}_p$, and 2 additions of eleventh degree polynomial whose coefficients are double precision integers. Below, this subsection shows only the polynomial product $u_0 = x_0 \cdot y_0$.

$$u_0 = u_{0,0} + u_{0,1}\gamma^3 + u_{0,2}\gamma^6,$$
$$u_{0,0} = x_{0,0} \cdot y_{0,0}, \quad u_{0,2} = x_{0,1} \cdot y_{0,1},$$
$$u_{0,1} = (x_{0,0} - x_{0,1})(y_{0,1} - y_{0,0}) + u_{0,0} + u_{0,2}. \qquad (9)$$

According to Eq. (9), the polynomial product requires 3 multiplications of cubic polynomial whose coefficients are single precision integers, 2 additions of cubic polynomial whose coefficients are elements in $\mathbb{F}_p$, and 2 additions of quintic polynomial whose coefficients are double precision integers. Below, this subsection shows only the polynomial product $u_{0,0} = x_{0,0} \cdot y_{0,0}$.

$$u_{0,0} = u_{0,0,0} + u_{0,0,1}\gamma + u_{0,0,2}\gamma^2 + u_{0,0,3}\gamma^3 + u_{0,0,4}\gamma^4,$$
$$u_{0,0,0} = x_{0,0,0} \cdot y_{0,0,0}, \qquad u_{0,0,4} = x_{0,0,2} \cdot y_{0,0,2},$$
$$v_0 = x_{0,0,1} \cdot y_{0,0,1}, \qquad v_1 = u_{0,0,0} + v_0,$$
$$u_{0,0,1} = (x_{0,0,0} - x_{0,0,1})(y_{0,0,1} - y_{0,0,0}) + v_0,$$
$$u_{0,0,2} = (x_{0,0,0} - x_{0,0,2})(y_{0,0,2} - y_{0,0,0}) + v_0 + u_{0,0,4},$$
$$u_{0,0,3} = (x_{0,0,1} - x_{0,0,2})(y_{0,0,2} - y_{0,0,1}) + v_1 + u_{0,0,4}. \qquad (10)$$

According to Eq. (10), the polynomial product requires 6 integer multiplications, 6 additions in $\mathbb{F}_p$, and 6 integer additions with double precision Therefore, the polynomial product $Z = X \cdot Y$ of Eq. (8) is finally given by

$$Z = u_0\gamma^2 + u_1\gamma^8 + u_2\gamma^{14} = \sum_{i=2}^{24} z_i\gamma^i$$
$$= (z_{14} - z_{13})\gamma + \sum_{i=2}^{11}(z_i + z_{i+13} - z_{13})\gamma^i + (z_{12} - z_{13})\gamma^{12},$$
$$\left(\because\ \gamma^{m+1=13} = 1, \ \sum_{i=1}^{m=12} \gamma^i = -1\right), \qquad (11)$$

where each z_i is a double precision integer. According to Eq. (11), the polynomial product additionally requires 22 integer multiplications with double precision. Last, by performing a reduction modulo p for every coefficient z_i of Z in Eq. (11), the product $Z = X \cdot Y$ in $\mathbb{F}_{p^{12}}$ can be obtained.

The case of the $\mathbb{F}_{p^4}$: Let 2 elements X and Y in $\mathbb{F}_{p^4}$ be represented by pseudo polynomial basis as

$$X = \sum_{i=0}^{m-1=3} x_i\gamma^{i+1}, \quad Y = \sum_{i=0}^{m-1=3} y_i\gamma^{i+1}, \quad x_i, y_i \in \mathbb{F}_p. \qquad (12)$$

Then, the product $Z = X \cdot Y$ in $\mathbb{F}_{p^4}$ is obtained as

$$
\begin{aligned}
&Z = \sum_{i=0}^{m-1=3} (v_i - v_4)\gamma^{i+1}, \quad \begin{aligned} x_{0,2} &= x_0 - x_2, & y_{0,2} &= y_2 - y_0, \\ x_{1,3} &= x_1 - x_3, & y_{1,3} &= y_3 - y_1, \end{aligned} \\
&u_0 = x_0 \cdot y_0, \quad u_1 = x_1 \cdot y_1, \quad u_2 = x_2 \cdot y_2, \quad u_3 = x_3 \cdot y_3, \\
&u_4 = (x_0 - x_1)(y_1 - y_0), \quad u_5 = (x_2 - x_3)(y_3 - y_2), \\
&u_6 = u_1 + u_3, \quad u_7 = x_{1,3} \cdot y_{1,3} + u_6, \quad v_0 = u_2 + u_7, \\
&u_8 = u_0 + u_2, \quad u_9 = x_{0,2} \cdot y_{0,2} + u_8, \quad v_3 = u_1 + u_9 \\
&v_4 = (x_{0,2} - x_{1,3})(y_{1,3} - y_{0,2}) + u_4 + u_5 + u_7 + u_9, \\
&v_1 = u_3 + u_5 + u_8, \quad v_2 = u_0 + u_4 + u_6. \qquad (13)
\end{aligned}
$$

This calculation cost is $9\tilde{M}_1 + 10A_1 + 18\tilde{A}_1 + 4R_1$.

3.5 Calculation Cost of Multiplication

Table 2 shows the calculation costs of a multiplication in the $\mathbb{F}_{p^{12}}$ constructed by type–$\langle h_{12}, m = 12\rangle$ GNB, and the $\mathbb{F}_{(p^3)^4}$ and $\mathbb{F}_{(p^4)^3}$ constructed by type–$\langle h_3, m = 3\rangle$ and type–$\langle h_4, m = 4\rangle$ GNBs. Note that this paper supposes that $h_3 = 2$, $h_4 = 1$, $h_{12} = 1$. Actually, these costs are comparable to those in *Optimal Extension Field* (OEF) [24, 25], which is often applied for the remarkable implementations of Ate–type pairing [22, 28]. According to **Table** 2, the implementations with Karatsuba's multiplication is faster than that with type–$\langle h, m\rangle$ CMVA. Thus, in what follows, this paper considers to adopt Karatsuba's multiplication.

In order to perform a multiplication in $\mathbb{G}_3$ by the calculation costs shown in **Table** 2, there must exist type–$\langle h_{12} = 1, m = 12\rangle$ GNB, or both type–$\langle h_3 = 2, m = 3\rangle$ and type–$\langle h_4 = 1, m = 4\rangle$ GNBs. Type–$\langle h = 1, m\rangle$ and type–$\langle h = 2, m\rangle$ GNBs, namely type–I and type–II ONBs, exist only when the following conditions are satisfied.

Condition 2 (Type–I ONB [26])

1. $n = m + 1$ is a prime number not equal to p.
2. The order of p in $\mathbb{F}_r$ is m.

Condition 3 (Type–II ONB [27])

1. $n = 2m + 1$ is a prime number not equal to p.
2. The order of p in $\mathbb{F}_r$ is $2m$, or the order of p in $\mathbb{F}_r$ is m and $2 \nmid m$.

Here, suppose that χ is assigned with a random integer. Then, according to **Table** 3, 4, 5, the theoretical probability to perform a multiplication by the calculation cost shown in **Table** 2 is about 81%. However, in order to make the security of the pairing–based cryptosystems maximum, it is desirable that the order r is a prime number. By considering to add this condition, the theoretical probability is reduced to about 60%.

4 Acceleration of Attack

In order to accelerate the rho method as **Alg**. 1, 2, the authors notes that the parameter required by the hash function η in **Alg**. 1 is only the 1–st element of an element in $\mathbb{G}_3$. Thus, the author considers that the rho method is improved to one which continues to compute only the 1–st element of the originally generated element in $\mathbb{G}_3$.

For instance, consider when $\mathbb{G}_3$ is the $\mathbb{F}_{p^{12}}$ constructed by type–$\langle h_{12} = 1, m = 12\rangle$ GNB. Let X, Y, and $Z = X \cdot Y$ in $\mathbb{F}_{p^{12}}$ be represented by pseudo polynomial basis as

$$
X = \sum_{i=0}^{m-1=12} x_i \gamma^{i+1}, \quad Y = \sum_{i=0}^{m-1=12} y_i \gamma^{i+1}, \quad Z = \sum_{i=0}^{m-1=12} z_i \gamma^{i+1},
$$
$$
x_i, y_i, z_i \in \mathbb{F}_p. \qquad (14)
$$

Then, the 1–st element in the product Z is obtained as

$$
\begin{aligned}
z_0 &= u_{0,11} + u_{1,10} + u_{2,9} + u_{3,8} + u_{4,7} + u_{5,6} \\
&\quad - u_{1,11} - u_{2,10} - u_{3,9} - u_{4,8} - u_{5,7} - x_0 y_0, \\
u_{i,j} &= (x_i - x_j)(y_i - y_j), \qquad (15)
\end{aligned}
$$

where each difference $(y_i - y_j)$ can be precomputed. Then, this calculation cost is given by $12\tilde{M}_1 + 22A_1 + 11\tilde{A}_1 + R_1$.

Here, let μ the function such that $\mu(X, Y) = z_0$, then a DLP on $\mathbb{G}_3$ can be solved by using **Alg**. 4. This algorithm iterates to compute the 1–st elements of the originally generated element in $\mathbb{G}_3$ N times, and then it completely computes the element in $\mathbb{G}_3$ after N steps. Actually, the above technique is called *tag tracing technique* [18], and this paper especially calls this walk *lazy random walk*. Note that this technique needs to precompute ${}_{T+1}\mathrm{H}_N (= {}_{T+N}\mathrm{C}_N)$ elements in $\mathbb{G}_3$ (namely all $V_{\mathbb{S}}$'s in $\mathbb{G}_3$) and to store the precomputed elements in the memory. Additionally, the acceleration technique with automorphism such as Frobenius map is not concurrently available for it. Here, suppose that the calculation costs of a multiplication and an addition in $\mathbb{Z}_r$ are respectively given by $\tilde{M}_1 + R_1$ and A_1. Then, the computation time of solving a DLP on $\mathbb{G}_3$ with each acceleration technique is shown in **Table** 6.

5 Experimentation

The authors experimented 128 DLPs in $\mathbb{G}_3$ with 62–bit prime numbers p and r, for which the computational environment shown in **Table** 1 is used. In this experimentation, Montgomery reduction [29] was applied for reductions modulo p. **Table** 6 shows the computation times of solving a DLP on $\mathbb{G}_3$. According to **Table** 6, compared to the automorphism technique, the proposed technique can be reduced the computation time by about 38% without occurrences of the fruitless cycles.

Algorithm 4: Solving with the rho method applied the tag tracing technique [18] optimized for the DLP on $\mathbb{G}_3$

```
Input: $X, Y\ (= X^c) \in \mathbb{G}_3$.
Output: $c \in \mathbb{Z}_r$.
1  for $i = 0$ to $T - 1$ do
2      assign random elements in $\mathbb{Z}_r$ to $a_i$, $b_i$.
3      $W_i \leftarrow X^{a_i} \cdot Y^{b_i}$, $w_i \leftarrow \nu(W_i)$.
4  $W_T \leftarrow 1$.
5  calculate $V_{\mathbb{S}}$ for every $\mathbb{S}$.
6  assign random elements in $\mathbb{Z}_r$ to $a_T$, $b_T$.
7  $W \leftarrow X^{a_T} \cdot Y^{b_T},\ w_T \leftarrow \nu(W)$.
8  for $i = T + 1$ to $r - 1$ do
9      $i \leftarrow i - 1,\ \mathbb{S} \leftarrow \{T, T, \ldots, T\}$.
10     for $l = 0$ to $N - 1$ do
11         $i \leftarrow i + 1,\ f \leftarrow w_{i-1} \pmod{T}$.
12         $a_i \leftarrow a_{i-1} + a_f,\ b_i \leftarrow b_{i-1} + b_f$.
13         $\mathbb{S} \leftarrow \mathbb{S} - \{T\} + \{f\}$.
14         if $i < N - 1$ then $w_i \leftarrow \mu(W, V_{\mathbb{S}})$.
15         else $W \leftarrow W \cdot V_{\mathbb{S}},\ w_i \leftarrow \nu(W)$.
16         if $w_i = w_j\ (0 \le j < i)$ then go to Step 17.
17 $c \leftarrow -(a_i - a_j)/(b_i - b_j)$.
```

† $\mathbb{S}$ denotes a repeated combination such that $\mathbb{S} = \{s_0, s_1, \ldots, s_{N-1}\}$ $(0 \le s_i \le T, s_i \le s_{i+1})$. Additionally, $V_{\mathbb{S}}$ denotes $V_{\mathbb{S}} = \prod_{i=0}^{N-1} W_{s_i}$.

‡ ν denotes the function such that $\nu(W) = w$ for $w \in \mathbb{F}_p$ which is the 1–st element of $W \in \mathbb{G}_3$.

Table 1: The computational environment

CPU	Intel Core i3–540 3.06 GHz (Only 1 core was used.)
Main memory	DDR3 PC3–10600 2.0 GB × 4
OS	Windows 7 Professional
Language	C
Compiler	GCC 4.6.2 (64–bit)
Compile Option	–O2

6 Conclusion

This paper demonstrated the efficiencies of the rho method with the automorphism technique and that with the tag tracing technique for solving DLP on $\mathbb{G}_3$ over BN curve. Additionally, it was shown by an experimentation that the proposed is certainly effective.

References

[1] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems Based on Pairings," SCIS 2000, pp. 26–28, 2000.

[2] D. Boneh and M. Franklin, "Identity–based Encryption from the Weil Pairing," Crypto 2001, Springer–Verlag, LNCS, Vol. 2139, pp. 213–229, 2001.

[3] D. Boneh, X. Boyan, and H. Shacham, "Short Group Signatures," Crypto 2004, Springer–Verlag, LNCS, Vol. 3152, pp. 41–55, 2004.

[4] I. F. Blake and A. C–F. Chan, "Scalable, Server–passive, User–anonymous Timed Release Cryptography," ICDCS 2005, pp. 504–523, 2005.

[5] D. Freeman, M. Scott, and E. Teske, "A Taxonomy of Pairing–Friendly Elliptic Curves," Journal of Cryptology, Vol. 23, Issue 2, pp. 224–280, 2010.

[6] P. S. L. M. Barreto and M. Naehrig, "Pairing–friendly Elliptic Curves of Prime Order," SAC 2005, Springer–Verlag, LNCS, Vol. 3897, pp. 319–331, 2006.

[7] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for Key Management: Part 1: General (Revision 3)," NIST Special Publication, Vol. 800–57 (Part 1), 2012.

[8] F. Hess, N. Smart, and F. Vercauteren, "The Eta–pairing Revisited," IEEE Trans. on Inf. Theory, Vol. 52, No. 10, pp. 4595–4602, 2006.

[9] E. Lee, H. S. Lee, and C. M. Park, "Efficient and Generalized Pairing Computation on Abelian Varieties," Cryptology ePrint Archive Report, No. 040, 2008.

[10] F. Vercauteren, "Optimal Pairings," IEEE Trans. on Inf. Theory, Vol. 56, No. 1, pp. 455–561, 2010.

[11] Y. Nogami, Y. Sakemi, H. Kato, M. Akane, and Y. Morikawa, "Integer Variable χ–based Cross Twisted Ate Pairing and Its Optimization for Barreto–Naehrig Curve," IEICE Trans. Fund., Vol. E92–A, No. 8, pp. 1859–1867, 2009.

[12] J. M. Pollard, "Monte Carlo Methods for Index Computation (mod p)," Mathematics of Computation, Vol. 32, No. 143, pp. 918–924, 1978.

[13] P. C. van Oorschot and M. J. Wiener, "Parallel Collision Search with Cryptanalytic Applications," Journal of Cryptology, Vol. 12, pp. 1–28, 1999.

[14] I. Duursma, P. Gaudry, and F. Morain, "Speeding up the Discrete Log Computation on Curves with Automorphisms," ASIACRYPT '99, Springer–Verlag, LNCS, Vol. 1716, pp. 103–121, 1999.

[15] K. Nekado, Y. Mori, T. Sumou, and Y. Nogami, "Representative Decision Efficient for Pollard's Rho Method on $\mathbb{G}_2$ over Barreto–Naehrig Curve," ITC–CSCC2012, No. F–W1–03, 2012.

[16] Y. Nogami, H. Kato, K. Nekado, S. Uehara, and Y. Morikawa, "Finding a Basis Conversion Matrix Using a Polynomial Basis Derived by a Small Multiplicative Cyclic Group," IEEE Trans. on Inf., Vol. 58, No. 7, pp. 4936–4947, 2012.

[17] S. Gao, “Abelian Groups, Gauss Periods and Normal Bases,” Finite Fields Appl. 7, No. 1, pp.148–164, 2001.

[18] J. H. Cheon, J. Hong, and M. Kim, “Accelerating Pollard’s Rho Algorithm on Finite Fields,” Journal of Cryptology, Vol. 25, Issue 2, pp. 195–242, 2012.

[19] E. Teske, “On Random Walks for Pollard’s Rho Method,” Mathematics of Computation, Vol. 70, No. 234, pp. 809–825, 2000.

[20] P. Wang and F. Zhang, “Computing Elliptic Curve Discrete Logarithms with the Negation Map,” Journal of Information Sciences, Vol. 195, pp. 277–286, 2012.

[21] D. Weber and T. Denny, “The Solution of McCurley’s Discrete Log Challenge,” Crypto 1998, Springer–Verlag, LNCS, Vol. 1462, pp. 458–471, 1998.

[22] D. F. Aranha, K. Karabina, P. Longa, C. H. Gebotys, and J. López, “Faster Explicit Formulas for Computing Pairings over Ordinary Curves,” Cryptology ePrint Archive Report, No. 526, 2010.

[23] K. Nekado, Y. Nogami, H. Kato, and Y. Morikawa, “Cyclic Vector Multiplication Algorithm and Existence Probability of Gauss Period Normal Basis,” IEICE Trans., Vol. E94–A, No. 1, pp. 172–179, 2011.

[24] D. Bailey and C. Paar, “Optimal Extension Fields for Fast Arithmetic in Public–Key Algorithms,” Asiacrypt 2000, Springer–Verlag, LNCS, Vol. 1976, pp. 248–258, 2000.

[25] A. Weimerskirch and C. Paar, “Generalizations of the Karatsuba Algorithm for Efficient Implementations,” Cryptology ePrint Archive Report, No. 224, 2006.

[26] Y. Nogami, A. Saito, and Y. Morikawa, “Finite Extension Field with Modulus of All–One Polynomial and Representation of Its Elements for Fast Arithmetic Operations,” IEICE Trans., Vol. E86–A, No. 9, pp. 2376–2387, 2003.

[27] Y. Nogami, S. Shinonaga, and Y. Morikawa, “Fast Implementation of Extension Fields with TypeII ONB and Cyclic Vector Multiplication Algorithm,” IEICE Trans., Vol. E88–A, No. 5, pp. 1200–1208, 2005.

[28] J.–L. Buchat, J. E. González–Díaz, S. Mitsunari, E. Okamoto, F. Rodríguez–Henríquez, and T. Teruya, “High–speed Software Implementation of the Optimal Ate Pairing over Barreto-Naehrig Curves,” Pairing 2010, Springer–Verlag, LNCS, Vol. 1976, pp. 248–258, 2000.

[29] P. L. Montgomery, “Modular multiplication without trial division,” Mathematics of Computation, Vol. 44, No. 170, pp. 519–521, 1985.

Table 2: The calculation cost of a multiplication in $\mathbb{G}_3$

Adopted Extension Field	$\mathbb{F}_{p^{12}}$		$\mathbb{F}_{(p^3)^4}$		$\mathbb{F}_{(p^4)^3}$	
Multiplication Algorithm	CVMA	Karatsuba	CVMA	Karatsuba[‡]	CVMA	Karatsuba[‡]
Calculation Cost [†]	$(78, 132, 77)$	$(54, 84, 150)$	$(60, 96, 87)$	$(54, 84, 108)$	$(60, 96, 78)$	$(54, 84, 132)$

[†] (i, j, l) denotes $i\tilde{M}_1 + jA_1 + l\tilde{A}_1 + 12R_1$. [‡] Type–$\langle h=2, m=3\rangle$ CVMA is adopted for each cubic extension part.

Table 3: Existence of type–$\langle h=1, m=12\rangle$ GNB

$\chi \pmod n$	0	1	2	3	4	5	6	7	8	9	10	11	12
$p \pmod n$	1	12	11	2	8	6	5	7	7	6	11	9	6
$r \pmod n$	1	6	0	0	3	12	10	12	0	1	9	11	0
Existence (Security*)	No	No	Yes (Low)	Yes (High)	No	Yes (High)	No	Yes (High)	Yes (Low)	Yes (High)	Yes (High)	No	Yes (Low)

Table 4: Existence of type–$\langle h=1, m=4\rangle$ GNB

$\chi \pmod n$	0	1	2	3	4
$p \pmod n$	1	3	3	3	4
$r \pmod n$	1	2	4	4	3
Existence (Security*)	No	Yes (High)	Yes (High)	Yes (High)	No

Table 5: Existence of type–$\langle h=2, m=3\rangle$ GNB

$\chi \pmod n$	0	1	2	3	4	5	6
$p \pmod n$	1	5	0	0	1	2	5
$r \pmod n$	1	6	4	2	3	6	6
Existence (Security*)	No	Yes (High)			No	Yes (High)	Yes (High)

* When $n \mid r$ (when r is certainly a composite number), this paper evaluates the security as "Low"–level one. When $n \nmid r$, this paper evaluates the security as "High"–level one.

Table 6: The computation cost and the number of iterations of random walk

		With automorphism	With tag tracing
The computation cost	With $\mathbb{F}_{p^{12}}$	$56\tilde{M}_1+86A_1+150\tilde{A}_1+14R_1$	$(14+42/N)\tilde{M}_1+(24+62/N)A_1+(11+139/N)\tilde{A}_1+(3+11/N)R_1$
	With $\mathbb{F}_{p^{(p^3)^4}}$	$56\tilde{M}_1+86A_1+108\tilde{A}_1+14R_1$	$(14+42/N)\tilde{M}_1+(34+50/N)A_1+(11+97/N)\tilde{A}_1+(3+11/N)R_1$
The number of iterations		average $\sqrt{\pi r/24}$	average $\sqrt{\pi r/2}$

Table 7: The experimental result when adopting the $\mathbb{F}_{p^{(p^3)^4}}$ as $\mathbb{G}_3$

	With automorphism	With tag tracing
The parameters	$T = 1024$	$T = 5$, $N = 7$
The number of iterations	average 6.1×10^7	average 15.5×10^7
The computation time	average 12 min. 11 sec.	average 7 min. 33 sec.