Solving 114-bit ECDLP for a Barreto-Naehrig Curve

Takuya Kusaka, Sho Joichi, Ken Ikuta, Md Al-Amin Khandaker, Yasuyuki Nogami, Satoshi Uehara, Nariyoshi Yamai, Sylvain Duquesne

Solving 114-bit ECDLP for a Barreto-Naehrig Curve

Takuya Kusaka[1], Sho Joichi[1], Ken Ikuta[1], Md. Al-Amin Khandaker[1], Yasuyuki Nogami[1], Satoshi Uehara[2], Nariyoshi Yamai[3], and Sylvain Duquesne[4]

[1] Graduate School of Natural Science and Technology, Okayama University, 700-8530, Okayama, Japan
{sho.joichi,pwu03iut,khandaker}@s.okayama-u.ac.jp
{t-kusaka,yasuyuki.nogami}@okayama-u.ac.jp
[2] Dept. of Information and Media Engineering, The University of Kitakyushu, 808-0135, Fukuoka, Japan
uehara@kitakyu-u.ac.jp
[3] Institute of Engineering, Tokyo University of Agriculture and Technology, 184-8588, Tokyo, Japan
nyamai@cc.tuat.ac.jp
[4] Univ Rennes, CNRS, IRMAR - UMR 6625, F-35000 Rennes, France
sylvain.duquesne@univ-rennes1.fr

Abstract. The security of cryptographic protocols which are based on elliptic curve cryptography relies on the intractability of elliptic curve discrete logarithm problem (ECDLP). In this paper, the authors describe techniques applied to solve 114-bit ECDLP in Barreto-Naehrig (BN) curve defined over the odd characteristic field. Unlike generic elliptic curves, BN curve holds an especial interest since it is well studied in pairing-based cryptography. Till the date of our knowledge, the previous record for solving ECDLP in a prime field was 112-bit by Bos et al in Certicom curve 'secp112r1'. This work sets a new record by solving 114-bit prime field ECDLP of BN curve using Pollard's rho method. The authors utilized sextic twist property of the BN curve to efficiently carry out the random walk of Pollard's rho method. The parallel implementation of the rho method by adopting a client-server model, using 2000 CPU cores took about 6 months to solve the ECDLP.

Keywords: ECDLP, Barreto-Naehrig curve, Pollard's rho method

1 Introduction

The complexity of solving a difficult mathematical problem such as discrete logarithm problem (DLP) and elliptic curve discrete logarithm problem (ECDLP) in a practical amount of time determines the security level of many popular public key cryptosystems. The mathematical estimation of such complexity is common in literature [10]. However, actual experiment is also bearing same importance, which is the focus of this work.

Pairing-based cryptography became popular as a form of public key cryptography (PKC) by the works of [18, 4, 3, 17]. Typically, pairing is defined as a bilinear map of two additive cyclic sub-groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order r to the same order multiplicative group $\mathbb{G}_3$. In practice, $\mathbb{G}_1$ and $\mathbb{G}_2$ are defined over a certain pairing-friendly curve of embedding degree k and $\mathbb{G}_3$ is defined in extension field $\mathbb{F}^*_{p^k}$. This paper considers Barreto-Naehrig curve [1], one of the widely used curves for pairing-based cryptographic applications. Let the curve be E, defined over the finite extension field $\mathbb{F}_{p^k}$, where embedding degree k is the smallest positive integer such that $r|(p^k - 1)$. The set of rational points $E(\mathbb{F}_p)$ is defined over the prime field $\mathbb{F}_p$, together with the *point at infinity* $\mathcal{O}$ forms a commutative group. The curve's order r is a large prime number such that $r|\#E(\mathbb{F}_p)$, where $\#E(\mathbb{F}_p)$ denotes the total number of rational points. The bilinearity property of pairing is e given as $e(aP, bQ) = e(P, Q)^{ab}$ for every rational point $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ and $a, b \in Z$. In the case of BN curve, $\mathbb{G}_1 = E(\mathbb{F}_p)$ and $\mathbb{G}_2 \subset E(\mathbb{F}_{p^{12}})$. The ECDLP in $\mathbb{G}_1 = E(\mathbb{F}_p)$ with two arbitrary rational point P, R is finding an integer s $(0 < s \approx r)$ such that $[s]P = R$ in E, provided that such s exists for P, R. The security of pairing-based cryptosystems depends on the difficulty of

- solving ECDLP in $\mathbb{G}_1$ and $\mathbb{G}_2$,
- solving DLP of the multiplicative group $\mathbb{G}_3$.
- and the difficulty of pairing inversion, i.e. $\mathbb{G}_1 \times \mathbb{G}_2 \leftarrow \mathbb{G}_3$.

This paper focuses on solving ECDLP in $\mathbb{G}_1 = E(\mathbb{F}_p)$ of BN curve.

The Pollard's rho algorithm [16] can solve the ECDLP in $\sqrt{\pi r/2}$ steps. Later Gallant et al. [8] improved the time complexity of the Pollard's rho method by $\sqrt{2}$ factors, that took $\sqrt{\pi r}/2$ steps. Van Oorschot and Wiener[20] proposed a distributed version of Pollard's rho algorithm that can be parallelized on n CPUs taking $\sqrt{\pi r}/2n$ steps giving n-fold speed up. This paper utilizes the above idea of parallelized rho method together with an efficient implementation of elliptic curve arithmetic using Montgomery reduction [13] and Montgomery trick [14].

In the experimental implementation, a client-server model consisting 2000 processor cores (each core acts as a single client for generating random rational points) is utilized. The generated random points with special feature also called the distinguished point, is sent to server to minimize collision detection cost. The server checks collision detection and uses MySQL database to store received rational points. The inversion in elliptic curve addition and doubling steps are efficiently carried out by applying Montgomery tricks while creating 95 random walks for each inversion in each thread. In addition, the sextic twist property of BN curve is utilized to apply skew Frobenius map on the twisted curve. The skew Frobenius map generates 6 associated rational points, which are treated as a single rational point in a random walk. The distinguished points technique is applied with the condition of 29 trailing zeros of x-coordinate to send the filtered point to the server. The experiment took about 6 months to solve the 114-bit ECDLP in BN curve defined over the prime field.

Related works

Several works have been done both from the algorithmic perspective and actual attack implementation. However, most of the works are focused on solving ECDLP in curves defined over binary field [2, 21]. A very few works have done on the actual attack for solving ECDLP in curves defined over $\mathbb{F}_p$. In 2016, Kajitani et al. [9] solved 70-bit prime field ECDLP in less than 24 minutes by using web-based volunteer computing. The authors of this paper have been influenced by Miyoshi et al. [12] work, where 94-bit ECDLP on was solved in 28 hours by parallelizing 71 computers. In 2012, Bos et al. [5] set a record by solving 112-bit ECDLP in prime field using about 200 Sony PlayStation 3s for about 6 months. Before that in 2002, the 109-bit ECDLP of Certicom ECCp-109 curve defined over $\mathbb{F}_p$ wast solved by Chris Monico [6] in 549 days of calendar time. Therefore, this works sets a new record for solving ECDLP on BN curve defined over 114-bit prime field.

2 Preliminaries

2.1 Elliptic Curve and ECDLP

Let p be a prime number and $\mathbb{F}_p$ be a prime field. An elliptic curve, generally represented by affine coordinates over $\mathbb{F}_p$ is defined as,

$$E(\mathbb{F}_p) : y^2 = x^3 + ax + b,\ a, b \in \mathbb{F}_p. \tag{1}$$

A pair of coordinates x and y that satisfy Eq.(1) are known as rational points on the curve. Let $\#E(\mathbb{F}_p)$ be the set of all rational points on the curve defined over $\mathbb{F}_p$ including the point at infinity denoted by $\mathcal{O}$.

Elliptic curve addition (ECA) between rational points $Q_1(x_1, y_1)$ and $Q_2(x_2, y_2)$ is $Q_1 + Q_2 = Q_3(x_3, y_3)$, defined as follows:

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{if } Q_1 \neq Q_2 \text{ and } x_1 \neq x_2, \\ \dfrac{3x_1^2 + a}{2y_1}, & \text{else if } Q_1 = Q_2 \text{ and } y_1 \neq 0, \\ \emptyset, & \text{otherwise,} \end{cases} \tag{2}$$

$$\big(x_3, y_3\big) = \begin{cases} \big(\lambda^2 - x_1 - x_2, (x_1 - x_3)\lambda - y_1\big), & \text{if } \lambda \neq \emptyset, \\ \mathcal{O}, & \text{otherwise.} \end{cases} \tag{3}$$

where Q_3 is also a rational points of elliptic curve $E(\mathbb{F}_p)$ and λ is the tangent between the points. If $Q_1 = Q_2$ then $Q_1 + Q_2 = 2Q_1$, which is known as elliptic curve doubling (ECD). ECDLP is the problem that calculates the scalar s only by using rational points P and Q in $E(\mathbb{F}_p)$ such that $Q = [s]P$, where $[s]P$ is

$$[s]P = \underbrace{P + P + \cdots + P}_{s-1 \text{ times addition of } P}. \tag{4}$$

2.2 BN Curve

BN curve is a class of non super-singular pairing friendly elliptic curve of embedding degree 12, defined over extension field $\mathbb{F}_q$ ($q = p^{12}$) is given by

$$E : y^2 = x^3 + b, \ (b \neq 0 \in \mathbb{F}_q \text{ and } x, y \in \mathbb{F}_q). \tag{5}$$

Other parameter settings are given by

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

$$r = 36\chi^4 - 36\chi^3 + 18\chi^2 - 6\chi + 1, \tag{7}$$

where χ is a certain integer and p is the characteristic of $\mathbb{F}_p$. Let $\#E(\mathbb{F}_q)$ be the set of all rational points on the curve defined over $\mathbb{F}_q$ including the point at infinity denoted by $\mathcal{O}$.

2.3 Pollard's Rho Method

Pollard's rho method is known as an efficient technique for solving an ECDLP. We designed and implemented a parallelized rho method for solving a 114-bit ECDLP with thousands of CPU cores. The proposed method consists of three steps. The first step generates a set of n different random rational points denoted by L. For an integer i where $1 \leq i \leq n$ and two random scalars α_i, β_i ($\in \mathbb{F}_q$), let W_i denote i-th random walk seed to be used in random walks where

$$W_i \triangleq [\alpha_i]P + [\beta_i]Q, \tag{8}$$

$$L \triangleq \{W_i | 1 \leq i \leq n\}. \tag{9}$$

Let m be the number of branches of the parallel random walks. The second step generates a set of m different random starting rational points in the m parallel random walk branches, denoted by U. For two integers i and j where $1 \leq i \leq m$ and $0 \leq j$, let $R_{i,j}$ denote the j-th rational point which is generated with random scalars $\alpha_{i,j}$ and $\beta_{i,j}$ by the i-th random walk branch. For an integer i where $1 \leq i \leq m$, since $R_{i,0}$ denotes the starting point of the i-th random walk branch where

$$R_{i,0} = [\alpha_{i,0}]P + [\beta_{i,0}]Q, \tag{10}$$

$R_{i,0}$ is randomly generated and,

$$U \triangleq \{R_{i,0} | 1 \leq i \leq m\}. \tag{11}$$

The third step performs m random works in parallel by using L and U. For a rational point R, let $\eta(R)$ be a function which gives an unique index of a rational point in L. For two integers i and j where $1 \leq i \leq m$ and $1 \leq j$, let the rational points $R_{i,j}$ which is the j-th rational point in the i-th random walk branch, be calculated by the following,

$$R_{i,j} = R_{i,j-1} + W_{\eta(R_{i,j-1})}, \tag{12}$$

$$= [\alpha_{i,j-1} + \alpha_{\eta(R_{i,j-1})}]P + [\beta_{i,j-1} + \beta_{\eta(R_{i,j-1})}]Q, \tag{13}$$

$$= [\alpha_{i,j}]P + [\beta_{i,j}]Q. \tag{14}$$

Let H be the set of all generated rational points in the rho method. Suppose the i-th random walk branch generate a rational point $R_{i,j} \in H$ at the j-th iteration step and we have a rational point $R_{i,j} = R_{i',j'}$ where $\alpha_{i,j} \neq \alpha_{i',j'}$ and $\beta_{i,j} \neq \beta_{i',j'}$, the case is called a collision and the ECDLP is solved by a simultaneous equation.

The algorithm of parallelized rho method is given as Alg.1. An example of parallelized rho method with 12-bit ECDLP is shown as follow Fig.1. In this paper, L and U are generated as preparation steps, and perform random walk at every random walk branches by using L and U. Each attacking clients run the algorithm. Therefore, the total number of random walk branches is m times the number of attacking clients. It is said that a collision would occur when $\sqrt{\pi r/2}$ points are generated on average according to the birthday paradox.

Algorithm 1: Parallelized rho Method

Input: $P,\ Q(=[s]P) \in E(\mathbb{F}_p)(0 \leq s < r)$
Output: s

```
1  for i = 1 to n do
2  |  α_i, β_i are random elements (0 ≤ α_i, β_i < r),
3  |  W_i ← [α_i]P + [β_i]Q.
4  |  H ← φ.
5  for i = 1 to m do
6  |  α_{i,0}, β_{i,0} are random elements (0 ≤ α_{i,0}, β_{i,0} < r),
7  |  R_{i,0} ← [α_{i,0}]P + [β_{i,0}]Q.
8  |  H ← H ∪ {R_{i,0}}.
9  for j = 1 to r − 1 do
10 |  for i = 1 to m do
11 |  |  l ← η(R_{i,j−1}).
12 |  |  R_{i,j} ← R_{i,j−1} + W_l, α_{i,j} ← α_{i,j−1} + α_l, β_{i,j} ← β_{i,j−1} + β_l.
13 |  |  if R_{i,j} = R_{i',j'}(R_{i',j'} ∈ H, α_{i,j} ≠ α_{i',j'}, β_{i,j} ≠ β_{i',j'}) then
   |  |  |  go to line 15.
14 |  |  else
   |  |  |  H ← H ∪ {R_{i,j}}.
15 s ← −(α_{i,j} − α_{i',j'}) / (β_{i,j} − β_{i',j'}) (mod r).
```

3 Techniques for accelerating random walk

In the attack, Montgomery reduction[13] is used for efficient modular arithmetics over $\mathbb{F}_p$. Montgomery trick[14] is also used for reducing the number of inversions over $\mathbb{F}_p$ by parallelizing many random walks on each client. Then, this paper attacks 114-bit ECDLP in $\mathbb{G}_1$ on BN curve to which a grouping technique is applied based on the sextic twist[15]. This section concentrates on introducing the grouping technique with skew Frobenius mapping defined in $\mathbb{G}_1$.

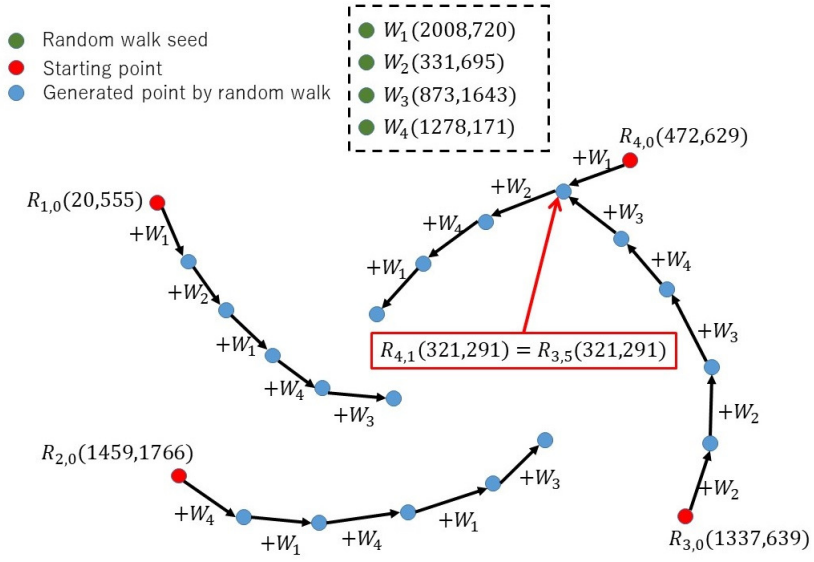

Fig. 1. parallelized rho method with 12bit ECDLP, n=4 and 4 branches

3.1 Groups of rational points for Ate pairing on BN curve

Let us remember the additive groups of rational points for Ate pairing e defined as follows, where $E(\mathbb{F}_{p^k})[r]$, Ker, and π denote the set of rational points of order r, kernel of homomorphism, and Frobenius mapping[7], respectively.

$$\mathbb{G}_1 = E(\mathbb{F}_{p^k})[r] \cap \mathrm{Ker}(\pi - [1]), \tag{15}$$

$$\mathbb{G}_2 = E(\mathbb{F}_{p^k})[r] \cap \mathrm{Ker}(\pi - [p]), \tag{16}$$

$$e(\cdot,\cdot) : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3 = \mathbb{F}^*_{p^k}/(\mathbb{F}^*_{p^k})^r. \tag{17}$$

In the case of BN curve, the embedding degree k is equal to 12 and the above $\mathbb{G}_1$ is equal to $E(\mathbb{F}_p)$. Based on these definitions, the next section introduces a grouping technique with sextic twist and skew Frobenius mapping.

3.2 Sextic twist and Skew-Frobenius Mapping

Basically, in order to improve Ate or optimal-ate pairing with BN curve, the sextic-twist technique is available. However, it also contributes to attacking the ECDLP on BN curve. Since the embedding degree of BN curve is 12 and BN curve is written as Eq. (5), sextic-twisted curve E' is given by

$$E' : \ y^2 = x^3 + bv^{-1}, \tag{18}$$

where v is a cubic and quadratic non residue in $\mathbb{F}_{p^2}$. In this case, we have the following isomorphism[19].

$$\mathbb{G}_1' = E'(\mathbb{F}_{p^{12}})[r] \cap \mathrm{Ker}(\pi^2 - [p^2]), \tag{19}$$

$$\begin{aligned}\psi_6 : \ & (x, y) \in \mathbb{G}_1 \\ & \longmapsto (v^{1/3}x, v^{1/2}y) \in \mathbb{G}_1'.\end{aligned} \tag{20}$$

$\mathbb{G}_1'$ has the following automorphism mapping $\tilde{\pi}$, where Q is a rational point in $\mathbb{G}_1$. It is called skew Frobenius mapping.

$$\begin{aligned}\tilde{\pi}(Q) &= \psi_6^{-1}(\pi^2(\psi_6(Q))) \\ &= (v^{\frac{p^2-1}{3}}x, v^{\frac{p^2-1}{2}}y).\end{aligned} \tag{21}$$

In this case, $\tilde{\pi}^6(Q) = \tilde{\pi}$, $v^{(p^2-1)/3}$ becomes a primitive cubic root of unity ϵ in $\mathbb{F}_p$, and $v^{(p^2-1)/2}$ becomes $p-1$. Thus, in what follows, the skew Frobenius $\tilde{\pi}$ map in this case is denoted by $\tilde{\pi}_6$ because it is periodic of period 6. Then, $\tilde{\pi}_6$ enables an efficient grouping in the rho method as introduced in the next section.

3.3 A grouping of rational points in $\mathbb{G}_1$ on BN Curve

Let us consider a rational point $T_i \in \mathbb{G}_1$ as generated in the random walk process. Then, based on the skew Frobenius mapping $\tilde{\pi}$, the following six rational points in $\mathbb{G}_1$ are easily obtained.

$$T_i = (x_i, y_i), \tag{22a}$$

$$\tilde{\pi}_6(T_i) = (\epsilon x_i, y_i), \tag{22b}$$

$$\tilde{\pi}_6^2(T_i) = (\epsilon^2 x_i, y_i), \tag{22c}$$

$$\tilde{\pi}_6^3(T_i) = (x_i, -y_i), \tag{22d}$$

$$\tilde{\pi}_6^4(T_i) = (\epsilon x_i, -y_i), \tag{22e}$$

$$\tilde{\pi}_6^5(T_i) = (\epsilon^2 x_i, -y_i). \tag{22f}$$

Then, a certain representative point among the six points is systematically and efficiently determined, which enables the following efficient grouping attack. For a rational point $R \in \mathbb{G}_1$, let $\mathrm{Rep}(R)$ denote a function which uniquely gives the representative in the group of six rational points given by the skew Frobenius map $\tilde{\pi}_6$.

Let us suppose that a collision is detected as

$$\mathrm{Rep}(T_i) = \mathrm{Rep}(T_j), \tag{23}$$

$$\mathrm{Rep}(T_i) = [\alpha_i]P + [\beta_i]Q, \tag{24}$$

$$\mathrm{Rep}(T_j) = [\alpha_j]P + [\beta_j]Q, \tag{25}$$

where $(\alpha_i, \beta_i) \neq (\alpha_j, \beta_j)$. Then, since $\tilde{\pi}_6^t(T_j) = [p^{2t}]T_j$, it means that the following relation holds:

$$\begin{aligned} T_i &= \tilde{\pi}_6^t(T_j), \\ T_i &= [p^{2t}]T_j, \\ [\alpha_i]P + [\beta_i]Q &= [p^{2t}]([\alpha_j]P + [\beta_j]Q), \\ \alpha_i + \beta_i \cdot s &\equiv p^{2t} \cdot \alpha_j + p^{2t} \cdot \beta_j \cdot s \pmod{r}, \\ s &\equiv -\frac{(\alpha_i - p^{2t} \cdot \alpha_j)}{(\beta_i - p^{2t} \cdot \beta_j)} \pmod{r}. \end{aligned} \tag{26}$$

where $0 \leq t < 6$. Therefore, since the skew Frobenius mapping is efficiently carried out as previously introduced, this grouping technique enables to reduce the average number for detecting a collision from $\sqrt{\pi r/2}$ to $\sqrt{\pi r/12}$. Alg. 2 accommodates the above procedure of obtaining representative point among the 6 associated rational points. The step 7, 12, 13 and 16 actually differs from Alg. 1. In this paper, the last 16 generated scalars α are saved in order to avoid fruitless cycles. If a new generated scalar α is same as one of the previous scalars, $\eta(R)$ in Sec.2.3 is incremented by 1.

4 Implementation

In this experiment, the authors employed about 2000 heterogeneous Intel64 CPU cores to attack the ECDLP by the rho method. In the parallel rho method, each random walk branches are executed on distributed computer resources in parallel. It is necessary to aggregate all the generated random rational points to a single computer in order to check the collision. Therefore, this paper employs a typical client-server model. In this context, all clients generate random rational points in parallel and only the distinguished points are sent to the server. The collision detection is done on the server.

4.1 Basic Integer Operations and Algebra

To handle the 114-bit integers efficiently, the authors employed 128-bit integer type on Intel64 CPUs. Since the target is a 114-bit ECDLP, an addition and a subtraction between two 114-bit integers never causes an overflow or an underflow. In contrast, since a result of a multiplication between two 114-bit integers easily exceeds 128-bit integer, the authors implemented big number arithmetic to handle this case.

In this attack, since the most of multiplications are multiplication modulo prime p or r; Montgomery reduction is implemented by using the 128-bit multiplication. A calculation of a multiplicative inverse is implemented by using the well-known extended Euclid's algorithm.

Since the majority of the computational cost of an ECA or an ECD is the cost of the inversion, well-known Montgomery trick is also employed. Since several

Algorithm 2: Customized parallelized rho method with representative point technique

Input: $P,\ Q(=[s]P) \in E(\mathbb{F}_p)(0 \leq s < r)$
Output: s

```
1  for i = 1 to n do
2      α_i, β_i are random elements (0 ≤ α_i, β_i < r),
3      W_i ← [α_i]P + [β_i]Q.
4      H ← φ.
5  for i = 1 to m do
6      α_{i,0}, β_{i,0} are random elements (0 ≤ α_{i,0}, β_{i,0} < r),
7      R_{i,0} ← Rep([α_{i,0}]P + [β_{i,0}]Q).
8      H ← H ∪ {R_{i,0}}.
9  for j = 1 to r − 1 do
10     for i = 1 to m do
11         l ← η(R_{i,j−1}).
12         if α_{i,j−1} = α_{i,c}(j − 18 < c < j − 1) then
               l++.
13         R_{i,j} ← Rep(R_{i,j−1} + W_l). α_{i,j} ← α_{i,j−1} + α_l, β_{i,j} ← β_{i,j−1} + β_l.
14         if R_{i,j} = R_{i',j'}(R_{i',j'} ∈ H, α_{i,j} ≠ α_{i',j'}, β_{i,j} ≠ β_{i',j'}) then
               go to line 16.
15         else
               H ← H ∪ {R_{i,j}}.
16 s ← −(α_{i,j} − p^{2t}·α_{i',j'}) / (β_{i,j} − p^{2t}·β_{i',j'}) (mod r).
```

inversions must be aggregated to utilize Montgomery trick, the authors evaluated the performance of our implementation of the inversion and choose 95 number of aggregation for Montgomery trick. Therefore, 95 random walk branches are synchronized and handled in a single thread in the implementation. The number of the threads in a client computer m is chosen as the number of real CPU cores or the number of Hyper Threading of the computer. For this case, the computational cost of a single step in rho method having Montgomery trick with a grouping (Sec.3.3) and without a grouping are shown in Table.1. The inversion takes 305 additions and a single multiplication on average.

Table 1. The computational cost of a single step in Rho method with Montgomery trick

Operations (mod p)	With grouping	Without grouping
Addition	1073	760
Multiplication	691	472
Squaring	95	95
Inversion	1	1

4.2 Distinguished Point Method

The collision detection is a check whether a rational point is in the set of all previously stored rational points or not. If the all generated rational points are sent to the server from the all clients, the number of rational points which must be stored in the server easily exceeds the capacity of the memory space of the server. For example, when solving 114-bit ECDLP, about $\sqrt{\pi \times 2^{114}/12} \approx 7.4 \times 10^{16}$ points need to be stored to the server. If the size of the data of a rational point is 50[Byte], 3.7×10^6[TB] storage space is required. Therefore, the authors choose distinguished point method to reduce the number of transmitted and stored rational points. In this attack, a distinguished point is a rational point where it's x-coordinate is divisible by an integer θ of the form of power of two. Here θ is called parameter of the distinguished point method. The computational cost to distinguish the point is a logical AND operation and comparison to numerical zero. The number of rational points sent to the server is reduced to $1/\theta$. The number of stored rational points are also reduced to $1/\theta$ which significantly reduce the cost of collision detection on the server. This reduction causes an overhead by generating extra rational points in the random walk branches.

If there is a pair of two rational points $(R_{i,j}, R_{i',j'})$ where $R_{i,j} = R_{i',j'}$ for positive integer h, $(\alpha_{i,j}, \beta_{i,j}) \neq (\alpha_{i',j'}, \beta_{i',j'})$, the following holds.

$$\eta(R_{i,j}) = \eta(R_{i',j'}), \tag{27}$$

$$R_{i,j+1} = R_{i',j'+1}, \tag{28}$$

$$R_{i,j+h} = R_{i',j'+h}. \tag{29}$$

In addition, if $\theta/2$ rational points are generated in a random walk branch, a distinguished point can be generated in the random walk branch on average. Therefore, the overhead is at most $\theta/2$ iterations in the pair of random walk branches, which means the distinguished method does not significantly increase the number of iteration steps before the rho method ends.

4.3 Aggregation on Generated Distinguished Points

By using the distinguished point method, the frequency to transmit the distinguished points can be significantly reduced. However, to reduce the frequency to transmit the IP datagrams, which includes the distinguished points, we employ a method to aggregate the distinguished points.

Each processor acts as a single client for generating random rational points in the experiment. The frequency to transmit the generated rational points to the server depends on the parameter of the distinguished method θ. A client stores a set of distinguished points. If the number of stored points becomes large enough, they are sent to the server at once. The number of the aggregated distinguished points is set 128 in this work. Therefore, the frequency to transmit the IP datagrams is reduced to 1/128 in the implemented system.

Table 2. Computer Resources

Client 1	The number of computers	150
	OS	Mac OSX 10.7.5 (64-bit)
	CPU	Intel Core i5 (2.50GHz)
Client 2	The number of computers	716
	OS	Windows7 Professional (64-bit)
	CPU	Intel Core 2 Duo E7600 (3.06GHz)
Client 3	The number of computers	162
	OS	FreeBSD 11 (64-bit)
	CPU	Dual Intel Xeon X5670 (2.90GHz)
Server	OS	CentOS 6.8 (64–bit)
	CPU	Intel Core i5 (3.40GHz)
	Database	MySQL ver. 5.1.73

4.4 Collision Detection at Server

The server stores information about coordinate of the rational point and scalars. For example, when the server receives $R(x,y) = [\alpha]P + [\beta]Q$, it stores x and α, β. When a new rational point is received, the server compares the information with stored rational points. If any corresponding point is found in the server, a scalar is calculated and the attack is ended.

By using Sec.4.2, the number of stored rational points is reduced. However, it is difficult to store all generated points on the memory of the server. Therefore, this study stores rational points on the MySQL databases.

4.5 Computer Resources

This attack employed several types of computers for both the server and the clients. Table 2 shows the major clients and the specification of the computers. The attack consists of 3 phases. The first phase was 51 days attack with Client 1 and 2. The second phase was 47 days attack with same clients. The final phase was 81 days attack with Client 3.

5 Result and Analysis

The parameter settings of BN curve for this attack is as follows:

$$\begin{aligned}
\chi &= 135000271, \\
b &= 3, \\
r &= 11957518425389075145185553763727233, \\
p &= 11957518425389075254535992784167879.
\end{aligned}$$

The rational points for this ECDLP attack were randomly chosen by Mersenne Twister (MT) pseudorandom number generator [11] as follows:

$$\begin{aligned} P = (&255580250207060501979974084777870,\\ &276621322973043076545206652160500 6),\\ Q = (&872919897487999939247623973935877 9,\\ &279749348673613711125158829029857 6). \end{aligned}$$

The x coordinate of rational point P is generated randomly where x satisfies Eq.(5) for the first time. The secret scaler s was incremented by 1 after generating randomly. The random seed of MT pseudorandom number generator is 5489. The Q is obtained as $[s]P = Q$. This s is only used to obtain Q in initial step but no longer used during the attack stage. After the 81 days, the attack introduced in Sec. 4.5 with 136887663 rational points stored in the server; ended with the following two pairs of random scalars of a collision:

$$\begin{aligned} (\alpha, \beta) = (&109781715530238573802933030283670 32,\\ &766718266516926106659451627955475 1),\\ (\alpha', \beta') = (&495683552457401174507454724116 32,\\ &144634401494496093146201363202877 3), \end{aligned}$$

where $[\alpha]P + [\beta]Q = [\alpha']P + [\beta']Q$. The secret scalar s was calculated by

$$\begin{aligned} s = &109286031977781172628425575559554 00/\\ &573667977516477501005305111620125 5. \end{aligned}$$

The results proved that this attack certainly succeeded to solve the 114-bit ECDLP. Note that 5.5GB of memory was consumed to store the information of generated rational points.

To save the memory space of the collision detection server and to reduce the network traffic, the authors disposed the information on the actual number of all the generated points, the authors give an estimation of the number. Since the authors chose $\theta = 2^{29}$ as the parameter of the distinguished points, we can estimate the the total number of generated rational points is $136887663 \times 2^{29} = 73491004476358656$. Since $r = 11957518425389075145185553763727233$, the result might be one of the typical average cases. However, the result of the authors attack is little bit better than the average number of rational points where a simple collision attack stops.

6 Conclusion

In this paper, the authors implemented a parallel rho attack for a 114-bit BN curve and solved an ECDLP for the curve. The attack system employed well-known Montgomery reduction, Montgomery trick and the extended Euclid' algorithm by using specially implemented integer operations over 128-bit integer

type on Intel64 CPUs. The authors also employed a grouping of rational points in $\mathbb{G}_1$ on BN curve which significantly reduced the number of useless collision detections in the attack. The results of the experiment indicates that the 114-bit ECDLP for BN curves can be solved in 81 days with 2000 cores of Intel Xeon X5670 (2.90GHz) CPU on average.

References

1. Barreto, P.S., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: International Workshop on Selected Areas in Cryptography, SAC 2005. pp. 319–331. Springer (2005)
2. Bernstein, D.J., Engels, S., Lange, T., Niederhagen, R., Paar, C., Schwabe, P., Zimmermann, R.: Faster elliptic-curve discrete logarithms on fpgas. Tech. rep., Cryptology eprint Archive, Report 2016/382 (2016)
3. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 506–522. Springer (2004)
4. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Advances in Cryptology—ASIACRYPT 2001, pp. 514–532. Springer (2001)
5. Bos, J.W., Kaihara, M.E., Kleinjung, T., Lenstra, A.K., Montgomery, P.L.: Solving a 112-bit prime elliptic curve discrete logarithm problem on game consoles using sloppy reduction. IJACT 2(3), 212–228 (2012), https://doi.org/10.1504/IJACT.2012.045590
6. Certicom: The Certicom ECC Challenge (Accessed: August 10, 2017), https://www.certicom.com/content/dam/certicom/images/pdfs/challenge-2009.pdf
7. Cohen, H., Frey, G., Avanzi, R., Doche, C., Lange, T., Nguyen, K., Vercauteren, F.: Handbook of elliptic and hyperelliptic curve cryptography. CRC press (2005)
8. Gallant, R., Lambert, R., Vanstone, S.: Improving the parallelized pollard lambda search on anomalous binary curves. Mathematics of Computation of the American Mathematical Society 69(232), 1699–1705 (2000)
9. Kajitani, S., Nogami, Y., Miyoshi, S., Austin, T., Al-Amin, K.M., Begum, N., Duquesne, S.: Web-based volunteer computing for solving the elliptic curve discrete logarithm problem. International Journal of Networking and Computing 6(2), 181–194 (2016)
10. Kim, T., Barbulescu, R.: Extended tower number field sieve: A new complexity for the medium prime case. In: Advances in Cryptology - CRYPTO 2016 - Proceedings, Part I. pp. 543–571. Springer (2016)
11. Matsumoto, M., Nishimura, T.: Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul. 8(1), 3–30 (Jan 1998), http://doi.acm.org/10.1145/272991.272995
12. Miyoshi, S., Nogami, Y., Kusaka, T., Yamai, N.: Solving 94-bit ecdlp with 70 computers in parallel. International Journal of Computer, Electrical, Automation, Control and Information Engineering 9(8), 1966 – 1969 (2015)
13. Montgomery, P.: Modular multiplication without trial division,. Math. Computation 44, 519–521 (1985)
14. Montgomery, P.L.: Speeding the pollard and elliptic curve methods of factorization. Mathematics of computation 48(177), 243–264 (1987)

15. Nogami, Y., Sakemi, Y., Okimoto, T., Nekado, K., Akane, M., Morikawa, Y.: Scalar multiplication using frobenius expansion over twisted elliptic curve for ate pairing based cryptography. IEICE transactions on fundamentals of electronics, communications and computer sciences 92-A(1), 182–189 (2009)
16. Pollard, J.M.: Monte carlo methods for index computation (mod p). Mathematics of computation 32(143), 918–924 (1978)
17. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 457–473. Springer (2005)
18. Sakai, R., Kasahara, M.: Id based cryptosystems with pairing on elliptic curve. IACR Cryptology ePrint Archive 2003, 54 (2003)
19. Sakemi, Y., Nogami, Y., Okeya, K., Kato, H., Morikawa, Y.: Skew frobenius map and efficient scalar multiplication for pairing–based cryptography. In: International Conference on Cryptology and Network Security. pp. 226–239. Springer (2008)
20. Van Oorschot, P.C., Wiener, M.J.: Parallel collision search with cryptanalytic applications. Journal of cryptology 12(1), 1–28 (1999)
21. Wenger, E., Wolfger, P.: Solving the discrete logarithm of a 113-bit koblitz curve with an fpga cluster. In: International Workshop on Selected Areas in Cryptography. pp. 363–379. Springer (2014)