Attack Experiments on Elliptic Curves of Prime and Binary Fields

Ni Ni Hla and Tun Myat Aung

Abstract At the beginning the paper describes the basic properties of finite field arithmetic and elliptic curve arithmetic over prime and binary fields. Then it discusses the elliptic curve discrete logarithm problem and its properties. We study the Baby-Step, Giant-Step method, Pollard's rho method and Pohlig–Hellman method, known as general methods that can exploit the elliptic curve discrete logarithm problem, and describe in detail attack experiments using these methods over prime and binary fields. Finally, the paper discusses the expected running time of these attacks and suggests the strong elliptic curves that are not vulnerable to these attacks.

1 Introduction

Elliptic Curve Cryptosystem (ECC) is an alternative approach for implementing Public Key Cryptosystem (PKC) in which each entity connecting in the public communication channel generally has a couple of keys, a public key and a private key to perform cryptographic operations such as encryption, decryption, signing, verification, and authentication. The private key must be kept secret but the corresponding public key is distributed to all entities connecting in the public communication channel [1]. ECC can be applied for providing the security services: confidentiality, authentication, data integrity, non-repudiation, and authenticated key exchange.

These days, ECC becomes a major in the industry of information and network security technology. It substitutes other public key cryptosystems such as RSA and DSA. It becomes the industrial standard as a consequence of an increase in speed and a decrease in power consumption during implementation as a result of less memory usage and smaller key sizes. Its security depends on the complexity of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP). Although the ECDLP is

N. N. Hla (✉) · T. M. Aung
University of Computer Studies, Yangon (UCSY), Yangon, Myanmar
e-mail: ni2hla@ucsy.edu.mm

T. M. Aung
e-mail: tma.mephi@gmail.com

A. Abraham et al. (eds.), *Emerging Technologies in Data Mining and Information Security*, Advances in Intelligent Systems and Computing 755,
https://doi.org/10.1007/978-981-13-1951-8_60

thought to be a difficult problem, it has not stopped attackers attempting to attack on ECC. Several attacks have been created, experienced and analyzed by mathematicians over the years, to discover defects in ECC. Some attacks have done partially well, but others have not.

The idea of this paper is to apply the knowledge of the general methods of attacking the ECDLP in attempting to select powerful elliptic curves over prime and binary fields under large integer. The structure of this paper is as follows. The Sect. 2 includes finite field arithmetic over prime and binary fields and their properties. In Sect. 3, we discuss elliptic curve arithmetic over prime and binary fields, its geometric properties, the ECDLP and its properties. The Sect. 4 describes in details the general methods of attacking on the elliptic curve discrete logarithm problem. In Sect. 5 we discuss attack experiments on the ECDLP over prime and binary fields. Finally, in Sect. 6 we conclude our discussion by describing time complexity of the attacking methods and by suggesting powerful elliptic curves for best secure implementation of ECC.

2 Finite Field Arithmetic

A finite field, generally signified by F, is a field which consists of a finite number of elements. Finite Fields are applied to the rational number system, the real number system and the complex number system. They contain a set of elements together with two arithmetic operations: *addition* signified by the symbol + and *multiplication* signified by the symbol, that satisfy the typical arithmetic properties:

- (F, +) is an *additive* group with operation by + and identity element by 0.
- (F\{0},) is a *multiplicative* group with operation by. and identity element by 1.
- Elements of finite group hold the distributive law: $(x + y) \cdot z = (x \cdot z) + (y \cdot z)$ for all $x, y, z \in F$

When the number of elements in the field is finite, then the field is said to be *finite* [2]. Galois open that the elements in the field to be finite and the number of elements should be p^m, where p is a prime number called the *characteristic* of the field and m is a positive integer. The finite fields are generally called *Galois fields* and also signified by $GF(p^m)$. When $m = 1$, then the field $GF(p)$ is called a *prime field*. When $m \geq 2$, then the field $GF(p^m)$ is called an *extension field*. The number of elements in a finite field is called the *order* of the field. Any two fields are *isomorphic* when their orders are the same [3].

2.1 Field Operations

Finite field F performs two arithmetic operations, *addition* and *multiplication*. However, the s*ubtraction* of field elements is defined in the expressions of addition operation. For instance, let $x, y \in F$, $x - y$ is defined as $x + (-y)$, in this case $-y$ is called *additive inverse* of *b* such that $y+(-y)=0$. Correspondingly, the *division* of field elements is defined in the expression of multiplication operation. For instance, let *x*, $y \in$ F with $y \neq 0$, *x/y* is defined as $x \cdot y^{-1}$, in this case y^{-1} is called the *multiplicative inverse* of *y* such that $y \cdot y^{-1} = 1$ [2].

Prime Field. A finite field of prime order *p* is called *prime field* signified by *GF(p)*. It contains a set of integer elements modulo *p*, $\{0,1,2,\ldots, p-1\}$ with additive and multiplicative groups performed modulo *p*. For any integer *x*, *x mod p* refers to the integer remainder *r* that obtained upon dividing *x* by *p*. This operation is called *reduction modulo p*. In this case, the remainder *r* is the distinct integer element between 0 and $p-1$, i.e. $0 \leq r \leq p-1$ [2]. The arithmetic operations of elements over *GF(p)* are performed as the following example (1).

Example (1). (*prime field* GF(23)) The elements of GF(23) are $\{0,1,2,\ldots,23\}$. The following examples demonstrate for arithmetic operations of elements in GF(23).

- Addition: $20+10 \pmod{23} = 7$ since $30 \bmod 23 = 7$.
- Subtraction: $10-20 \pmod{23} = 13$ since $10 + (-20) \bmod 23 = 13$.
- Multiplication: $20 \cdot 10 \pmod{23} = 6$ since $200 \bmod 23 = 6$.
- Inversion: $10^{-1} \pmod{23} = 7$ since $10 \cdot 7 \bmod 23 = 1$.
- Division: $20/10 \pmod{23} = 2$ since $20.\ 10^{-1} \pmod{23}$ and $20.\ 7 \pmod{23} = 2$.

Binary Field. A finite field of order 2^m is called *binary field* signified by $GF(2^m)$. It also refers to the *finite field with characteristic-two*. The elements over $GF(2^m)$ can be constructed by applying a *polynomial basis representation* defined by the Eq. (1). In this case, the elements of $GF(2^m)$ are the binary representation polynomials with degree at most $m-1$.

$$GF(2^m) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_2x^2 + a_1x + a_0,\ a_i \in \{0, 1\}. \tag{1}$$

f (x) is defined as an irreducible binary representation polynomial with degree *m* if *f(x)* cannot be factored as a product of binary representation polynomials with degree less than *m*. Let *a(x)* and *b(x)* be elements over $GF(2^m)$. They are the binary representation polynomials with degree at most $m-1$. Addition of elements in binary field refers to the addition of binary representation polynomials, that is,$a(x) \oplus b(x)$. Multiplication of elements in $GF(2^m)$ refers to refers to the expression $a(x) \times b(x) \bmod f(x)$. Let $c(x) = a(x) \times b(x)$ and *c(x)* be an binary representation polynomial with degree more than *m*. The result of the expression *c(x)* mod *f(x)* refers to the unique remainder polynomial *r(x)* with degree less than *m* that obtained upon the division of *c(x)* by *f(x)*; this operation is called *reduction modulo f(x)*. Division of elements in $GF(2^m)$ refers to refers to the expression $a(x)/b(x) \bmod f(x)$. In this case, the division of elements in $GF(2^m)$ is calculated as the expression $a(x) \times b(x)^{-1} \bmod f(x)$ [2]. The arithmetic operations of elements over *GF(2 ^{m})* are performed as the following example (2).

Example (2). (*binary field* $GF(2^m)$) The elements of $GF(2^m)$ generated by the polynomial $f(x) = x^4 + x + 1$ are represented by 16 binary polynomials of degree at most 3 as shown in Table (1).

Table 1 Binary representation polynomials

Polynomial	Polynomial	Polynomial	Polynomial
0	x^2	x^3	$x^3 + x^2$
1	$x^2 + 1$	$x^3 + 1$	$x^3 + x^2 + 1$
x	$x^2 + x$	$x^3 + x$	$x^3 + x^2 + x$
$x + 1$	$x^2 + x + 1$	$x^3 + x + 1$	$x^3 + x^2 + x + 1$

The followings are some examples of arithmetic operations in $GF(2^4)$ with the elements generated by reduction polynomial $f(x) = x^4 + x + 1$.

- Addition: $(x^3 + 1) + (x + 1) = x^3 + x$ since $(x^3 + 1) \oplus (x + 1) = x^3 + x$.
- Subtraction: $(x^3 + 1) - (x + 1) = x^3 + x$ since $(x^3 + 1) \oplus (x + 1) = x^3 + x$.
- Multiplication $(x^3 + 1).(x + 1) = x^3$ since $(x^3 + 1) \times (x + 1) = x^4 + x^3 + x + 1$ and $(x^4 + x^3 + x + 1) \bmod f(x) = x^3$.
- Inversion: $(x + 1)^{-1} = x^3 + x^2 + x$ since $(x^3 + x^2 + x) \times (x + 1) \bmod f(x) = 1$.
- Division: $(x^3+1)/(x+1) = x^2+x+1$ since $(x^3+1)\times(x+1)^{-1} \bmod f(x) = x^2+x+1$

3 Elliptic Curve Arithmetic

The elliptic curve over finite field E(GF) is a cubic curve defined by the general Weierstrass equation: $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ over GF where $a_i \in GF$ and GF is a finite field. We study elliptic curves over $GF(p)$ and $GF(2^m)$.

3.1 Elliptic Curve Arithmetic Over Prime Field -GF(P)

Elliptic curves are driven from the general Weierstrass equation. The elliptic curve E(GF(p)) over $GF(p)$ is determined by the Eq. (2) [4]:

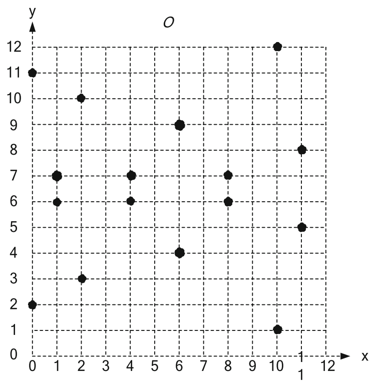
$$y^2 = x^3 + ax + b, \tag{2}$$

where p>3 is a prime and $a, b \in GF(p)$ satisfy that $4a^3 + 27b^2 \neq 0$. ($a_1 = a_2 = a_3 = 0$; $a_4 = a$ and $a_6 = b$ corresponding to the general Weierstrass equation)

Points on E(GF(p)). The elliptic curve E(GF(p)) over GF(p) belongs to a set of points together with a point at infinity signified by symbol O. In this case,$\{P = (x, y)|y^2 = x^3 + ax + b; x, y, a, b \in GF(p)\}$. Every point on the curve generally has its corresponding *inverse*. The inverse of a point (x, y) on E(GF(p)) is defined as $(x, -y)$. The number of points on the curve, including a point at infinity, is defined as its *order #E*. The points on E(GF(p)) are generated by using Algorithm (1).

(0, 2)	(0, 11)
(1, 6)	(1, 7)
(2, 3)	(2, 10)
(4, 6)	(4, 7)
(6, 4)	(6, 9)
(8, 6)	(8, 7)
(10, 1)	(10, 12)
(11, 5)	(11, 8)

(a) Points



(b) Graph

Fig. 1 Points on $E : y^2 = x^3 + 5x + 4$

Algorithm (1). Generating the points on E(GF(p))

```
Input: a, b, p
Output: $P_i = (x_i, y_i)$
Begin
x = 0;
while( x < p ){
    $w = (x^3 + ax + b) \bmod p$.
    If(w is perfect square in $Z_p$) output $(x, \sqrt{w}), (x, -\sqrt{w})$
    x = x + 1.
}
End
```

Example (3). Let p = 13 and consider the elliptic curve $E : y^2 = x^3 + 5x + 4$ over GF(13) where a=5 and b=4. Note that $4a^3 + 27b^2 = 500 + 432 = 932 \bmod 13 = 9$, so E is indeed an elliptic curve. The points on the curve and its graph are shown in Fig. (1a and b). The *order* of the elliptic curve $E : y^2 = x^3 + 5x + 4$ over GF(13) is 17.

Arithmetic Operations on E(GF(p)). Addition of two points on an elliptic curve E(GF(p)) applied the *chord-and-tangent rule* to find a third point on the curve. The addition operation with the points on E(GF(p)) generates a group with point at infinity O serving as its identity. It is the group of points on E(GF(p)) that is used in the construction of elliptic curve cryptosystems [5]. It is the best way to explain the point addition rule geometrically. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two distinct points on E(GF(p)). Assume that the point $R = (x_3, y_3)$ is obtained by *addition* of P and Q. This point addition is illustrated in Fig. (2a). The line connecting through P and Q intersects the elliptic curve at the point called -R. R is the reflection of -R with respect to the x-axis. Assume that *doubling* of P is $R = (x_3, y_3)$ in the case of $P = (x_1, y_1)$. This point doubling is illustrated in Fig. (2b). The tangent line drawing from point P intersects the elliptic curve at the point called -R. R is the reflection of -R with respect to the x-axis as in the case of addition.

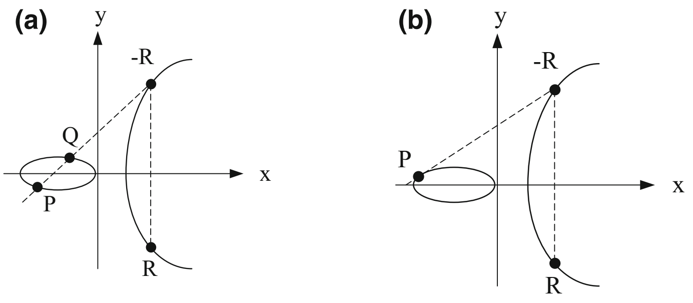


Fig. 2 **a** Addition. ($R = P + Q$). **b** Doubling. ($R = P + P$)

The geometric description open following algebraic methods for the addition of two points and the doubling of a point [4].

1. $P + O = O + P = P$ for all $P \in E(GF(p))$.
2. If $P = (x, y) \in E(GF(p))$, then $(x, y) + (x, -y) = o$ where the point (x, -y) is signified by (-P) that is called the *inverse* of P.
3. (*Point addition*). Let $P = (x_1, y_1) \in E(GF(p))$ and $Q = (x_2, y_2) \in E(GF(p))$, where $P \neq \pm Q$. Then $P + Q = (x_3, y_3)$, where $x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = \lambda(x_1 - x_3) - y_1$. In this case, $\lambda = (y_2 - y_1)/(x_2 - x_1)$.
4. (*Point doubling*). Let $P = (x_1, y_1) \in E(GF(p))$, where $P \neq -P$. Then $2P = (x_3, y_3)$, where $x_3 = \lambda^2 - 2x_1$ and $y_3 = \lambda(x_1 - x_3) - y_1$. In this case, $\lambda = (3x_1^2 + a)/2y_1$.

Example (4). (*elliptic curve addition and doubling*) Let us consider the elliptic curve defined in Example (3).

- *Addition*. Let P = (1, 6) and Q = (4, 6). Then P + Q = (8, 7).
- *Doubling*. Let P = (1, 6). Then 2P = (10, 1).
- *Inverse*. Let P = (1, 6). Then –P = (1, 7).

3.2 Elliptic Curve Arithmetic Over Binary Field - GF(2 ^{M})

Elements over GF(2 ^{m}) must be firs generated by using a reduction polynomial f(x). These elements are applied to construct an elliptic curve E(GF(2 ^{m})) over GF(2 ^{m}). The curve E(GF(2 ^{m})) is determined by the Eq. (3) [4]:

$$y^2 + xy = x^3 + ax + b. \tag{3}$$

where $a, b \in GF(2^m)$ and $b \neq 0$.

Points on E(GF(2 ^{m})). The elliptic curve E(GF(2 ^{m})) over GF(2 ^{m}) belongs to a set of points together with a point at infinity signified by symbol O. In this case,

Table 2 Binary and polynomial representations for elements of GF(2^4)

Binary	Polynomial	Binary	Polynomial	Binary	Polynomial	Binary	Polynomial
0000	0	0100	x^2	1000	x^3	1100	x^3+x^2
0001	1	0101	x^2+1	1001	x^3+1	1101	x^3+x^2+1
0010	x	0110	x^2+x	1010	x^3+x	1110	x^3+x^2+x
0011	$x+1$	0111	x^2+x+1	1011	x^3+x+1	1111	x^3+x^2+x+1

$\{P = (x, y)|y^2 + xy = x^3 + ax + b; x, y, a, b \in GF(2^m)\}$. Every point on the curve has its corresponding *inverse*. The inverse of a point (x, y) on E(GF(2 ^{m})) is defined as $(x, x \oplus y)$. The number of points on the curve, including a point at infinity, is generally called its *order #E*. The points on E(GF(2 ^{m})) are generated by using Algorithm (2).

```
Algorithm (2). Generating the points on E(GF(2^m))
Input: a, b, f(x)
Output: $P_i = (x_i, y_i)$
Begin
$x_i = \{0,1,g^1,K,g^{m-2}\}$
$y_i = \{0,1,g^1,K,g^{m-2}\}$

for(i=0; i<2^m; i++){
    for(j=0; j < 2^m ;j++){
     $w_1 = x_i^3 \oplus ax_i \oplus b$ .
     $w_2 = y_j^2 \oplus x_i y_j$
    If ($w_1 = w_2$) output $(x_i, y_j)$ , $(x_i, y_j \oplus x_i)$
    }
}
End
```

Example (5). Let $f(x) = x^4 + x + 1$ be the reduction polynomial. Then binary and polynomial representations for 16 elements of GF(2^4) generated by the reduction polynomial $f(x) = x^4 + x + 1$ are shown in Table (2).

Table (3) shows the power representations of g and corresponding binary representations for elements of GF(2^4) generated by the reduction polynomial $f(x) = x^4 + x + 1$. The element of g = (0010) is a generator of GF(2^4) and its order is 15 (2^4 – 1).

The elliptic curve $E : y^2 + xy = x^3 + g^{11}x + g^{13}$ where $a = g^{11}$ and $b = g^{13}$ belongs to the points on the curve, as shown in Fig. (3). The points on the curve and its graph are shown in Fig. (3a and b). The *order* of the elliptic curve $E : y^2 + xy = x^3 + g^{11}x + g^{13}$ is 22.

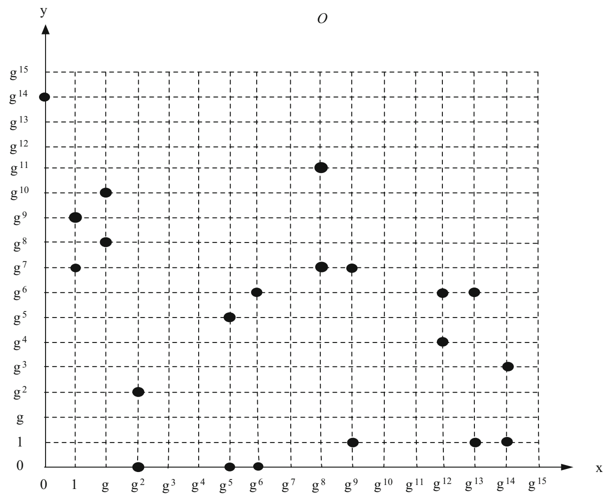
Arithmetic Operations on E(GF(2 ^{m})). Addition of two points on an elliptic curve E(GF(2 ^{m})) also applied the *chord-and-tangent rule* to find a third point on the curve. The addition operation with the points on E(GF(2 ^{m})) generates a group

Table 3 Power and binary representations of elements of GF(2^4)

Power	Binary	Power	Binary	Power	Binary	Power	Binary
g	0010	g^5	0110	g^9	1010	g^{13}	1101
g^2	0100	g^6	1100	g^{10}	0111	g^{14}	1001
g^3	1000	g^7	1011	g^{11}	1110	g^{15}	0001
g^4	0011	g^8	0101	g^{12}	1111		

$(0,g^{14})$	O
$(1,g^7)$	$(1,g^9)$
(g,g^8)	(g,g^{10})
$(g^2,0)$	(g^2,g^2)
$(g^5,0)$	(g^5,g^5)
$(g^6,0)$	(g^6,g^6)
(g^8,g^7)	(g^8,g^{11})
$(g^9,1)$	(g^9,g^7)
(g^{12},g^4)	(g^{12},g^6)
$(g^{13},1)$	(g^{13},g^6)
$(g^{14},1)$	(g^{14},g^3)

(a) Points



(b) Graph

Fig. 3 Points on $E : y^2 + xy = x^3 + g^{11}x + g^{13}$

with point at infinity O serving as its identity. It is the group of points on E(GF(2 ^{m})) that is used in the construction of elliptic curve cryptosystems [5]. The followings are algebraic methods for the addition of two distinct points and the doubling of a point [4].

1. $P + O = O + P = P$ for all $P \in E(GF(2^m))$.
2. If $P = (x, y) \in E(GF(2^m))$, then $(x, y) + (x, x + y) = O$ where the point (x, $x+y$) signified by (-P) is called the *inverse* of P.
3. (*Point addition*). Let $P = (x_1, y_1) \in E(GF(2^m))$ and $Q = (x_2, y_2) \in E(GF(2^m))$, where $P \neq \pm Q$. Then $P + Q = (x_3, y_3)$, where $x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$ and $y_3 = \lambda(x_1 + x_3) + x_3 + y_1$. In this case, $\lambda = (y_2 + y_1)/(x_2 + x_1)$.
4. (*Point doubling*). Let $P = (x_1, y_1) \in E(GF(2^m))$, where $P \neq -P$. Then $2P = (x_3, y_3)$, where $x_3 = \lambda^2 + \lambda + a$ and $y_3 = x_1^2 + \lambda x_3 + x_3$. In this case, $\lambda = x_1 + (y_1/x_1)$.

Example (6). (*elliptic curve addition and doubling*) Let's consider the elliptic curve defined in Example (5).

- *Addition*. Let $P = (g^2, g^2)$ and $Q = (g^6, g^6)$. Then $P + Q = (g^5, 0)$.
- *Doubling*. Let $P = (g^2, g^2)$. Then $2P = (g^{14}, 1)$.
- *Inverse*. Let $P = (g^2, g^2)$. Then $-P = (g^2, 0)$.

3.3 Elliptic Curve Discrete Logarithm Problem

The complexity of solving ECDLP determines the security of ECC. Let P and Q be the points on an elliptic curve such that $Q{=}kP$, where k is an integer number. k is called the discrete logarithm of Q to the base P. Known two points, P and Q, it is unable to compute k, when the group order of the points is enough large.

Point Multiplication. *Point Multiplication* is a major operation usually used in ECC. The scalar multiplication operation of a integer scalar k with a point P on the elliptic curve creates another point Q on this curve [1]. The point Q is gotten by performing *point addition and point doubling* operations according to bit sequence patterns of integer scalar k. The bit sequence patterns of integer k is shown as the Eq. (4)

$$k = k_{n-1}2^{n-1} + k_{n-2}2^{n-2} + n + k_1 + k_0, \tag{4}$$

where $k_{n-1} = 1$ and $k_i \in \{0, 1\}, i = 0, 1, 2, \ldots, n-1$. This operation is based on the *binary method* which scans the bit sequence patterns of k either from left-to-right or right-to-left [2]. The Algorithm-3 illustrates the scalar multiplication operation of a integer scalar k with a point P on the elliptic curve using binary method. This method can be applied for both elliptic curves over GF(p) and GF(2 ^{m}).

```
Algorthm (3). Scalar Multiplication of a Point
Input : point P and integer scalar k
Output : point Q such that Q = kP
Begin
$k_i \in \{0,1\}, i = 0,1,2,\ldots,n-1$
Q=P
For i = n-1 to 0 do
{
   Q = Point-Doubling of Q
   If $k_i$ = 1 then
   Q = Point-Addition of P and Q
}
Return Q
End
```

The cost of this point multiplication method relies on the number of 1 s in bit sequence patterns of integer scalar k. The number of 1 s is called the *Hamming Weight* of the scalar. Generally, this method performs (n − 1) *point doublings* and (n − 1)/2 *point additions*. For each bit .1., this method must perform both *point doubling* and *point addition*, if the bit is .0., this method performs only *point doubling* operation.

Consequently, making less the number of 1 s in bit sequence patterns of integer scalar k will increase the speed of scalar multiplication of a point on elliptic curve [6].

The Order of Point. Let $P \in E(GF(p))$. The *order* of point P is defined as the smallest positive integer value N such that $NP = O$. In this case, O is the identity of the group of points on the elliptic curve.

$$p + 1 - 2\sqrt{p} \leq N \leq p + 1 + 2\sqrt{p}. \tag{5}$$

All different values of N must be tried in the range defined in the Eq. (5) [7] and then check which value of N agrees this statement $NP = O$.

Example (7). Let E be the elliptic curve $E : y^2 = x^3 + 5x + 4$ over GF(13). The order of point $(1, 7)$ is 17. The Eq. (5) is solved as $13+1-2\sqrt{13} \leq N \leq 13+1+2\sqrt{13}$. All different values of N in the range, $7 \leq N \leq 21$, could be tried to find $N = 17$ such that $NP = O$. Therefore, $N = 17$.

4 General Methods of Attacking on ECDLP

The complexity of solving the Discrete Logarithm Problem (DLP) is deeply important for the security of PKC. PKC is constructed based on the assumption that the DLP is extremely difficult to compute; the more difficult it is, the more security it supports. Therefore, PKC is constructed on a larger group order under large integer in order to increase the complexity of solving the DLP.

General methods of attacking on the ECDLP can be classified into three groups as following [8]. These methods can solve the ECDLP under small integer.

1. Methods stand on random walks, such as the exhaustive search method and the Baby-Step, Giant-Step method,
2. Methods stand on random walks with special conditions, like Pollard's rho method and Pollard's lambda method, and
3. Methods stand on multiplicative groups, such as the Index Calculus method and Pohlig–Hellman method.

We studied the following general methods of attacking on the ECDLP.

4.1 Baby-Step, Giant-Step Method

Let $P, Q \in E$. Assume that we solve an integer scalar k such that $Q = [k]P$ and P has prime order N. At first, we must compute the order N of P. This method generally performs about $\sqrt{N}$ steps and requires about $\sqrt{N}$ storage. Therefore, this method only works well for memory storage size N. This method follows the procedure below [7].

1. Fix an integer m such that $m = \left\lceil \sqrt{N} \right\rceil$ and compute mP.
2. Compute and store a list of iP for $1 \leq i \leq m$.
3. Compute the points such that $Q - jmP$ for $j = 0, 1, \ldots$ until one of resulting points matches one from the stored list.
4. If $iP = Q - jmP$, then $Q = kP$ with $k \equiv i + jm (\bmod N)$.

The list of points iP are calculated by adding P to $(i-1)P$. It is *the baby-step*. The list of points $Q - jmP$ are computed by adding $-mP$ to $Q - (j-1)mP$. It is *the giant-step*. This method may generally perform about m steps to find a match and its time complexity is $O\left(\sqrt{N}\right)$ [7].

4.2 Pollard's Rho Method

Let $P, Q \in E$. Assume that we solve an integer scalar k such that $Q = [k]P$ where P has prime order N and $Q \in < P >$. This method generally find two different pair of integers: (a, b) and $\left(a', b'\right)$ modulo N such that $[a]P + [b]Q = [a']P + [b']Q$. This method follows the procedure below [7]:

1. Select $a, b \in [0, N-1]$ uniformly at random.
2. Compute $[a]P + [b]Q$.
3. Store the triple $(a, b, [a]P + [b]Q)$.
4. Select new pairs $\left(a', b'\right)$ uniformly at random such that $(a, b) \neq \left(a', b'\right)$.
5. Compute $[a']P + [b']Q$.
6. Store the new triple $(a', b', [a']P + [b']Q)$.
7. Compute and Check the new triple against all previously stored triples until we find a pair $\left(a', b'\right)$ satisfied with the Eq. (6).
8. Compute $k \equiv (a - a')(b' - b)^{-1} \bmod N$.

$$[a]P + [b]Q = [a']P + [b']Q \tag{6}$$

The time complexity of this method is $O\left(\sqrt{\pi N}/2\right)$ [7]. The diagram of the sequence of resulting points looks like the Greek letter ρ. Therefore, this method is called the Pollard-Rho method.

4.3 Pohlig–Hellman Method

Let $P, Q \in E$. Assume that we solve an integer scalar k such that $Q = [k]P$ where P has prime order N.

$$N = \prod\nolimits_i q_i^{e_i} \tag{7}$$

The main idea of this method is as following:

- Compute the order N of P.
- Compute prime factorization of N that satisfied the Eq. (7).
- Compute $k(\text{mod} q_i^{e^i})$ for each i,
- Combine them to obtain k (mod N) using the Chinese Remainder theorem [9].

Let q be a prime, and let q^e be the exact power of q dividing N. This method defines k in its base q expansion as the Eq. (8).

$$k = k_0 + k_1 q + k_2 q^2 + n \tag{8}$$

where $0 \le k_i < q$. This method evaluates $k(\text{mod} q_i^e)$ by successively determining $k_0, k_1, k_2, n, k_{e-1}$. This method follows the procedure below [7]:

1. Compute $T = j.\left(\frac{N}{q}.P\right)$, 0≤j≤q-1.
2. Compute $\frac{N}{q}.Q$. It is an element of $k_0\left(\frac{N}{q}.P\right)$ of T.
3. If $e = 1$, stop. Otherwise, continue.
4. Let $Q_1 = Q - k_0 P$.
5. Compute $\frac{N}{q^2}.Q$. It is an element of $k_1\left(\frac{N}{q^2}.P\right)$ of T.
6. If $e = 2$, stop. Otherwise, continue. Assume that we have calculated: k_1, k_2, n, k_{r-1} and Q_1, Q_2, n, Q_{r-1}.
7. Let $Q_r = Q_{r-1} - k_{r-1} q^{r-1} P$.
8. Determine k_r such that $\frac{N}{q^{r+1}}.Q_r = k_r\left(\frac{N}{q}.P\right)$.
9. If $r = e - 1$, stop. Otherwise, return to step (7).

Then the method computes $k \equiv k_0 + k_1 q + n + k_{e-1} q_{e-1} (\text{mod} q_e)$. Therefore, early we find k_1. In the same way, the method produces k_2, $k_{3,n}$.We must stop after $r = e - 1$. The time complexity of this method is $O\left(\sqrt{q}\right)$ [7]. In this case, q is the largest prime divisor of N. In practice this method becomes infeasible as a result of that N has a large prime divisor. Then it becomes difficult to make and store the list T to find matches.

5 Attack Experiments

We implemented well-known general common attacks such as Baby-Step Giant-Step method, Pollard's rho method and the Pohlig–Hellman method by using our implementations of finite field arithmetic operations [10] and elliptic curve arithmetic operations [11] under java BigInteger class.

5.1 Baby-Step Giant-Step Attack

Prime Field. Let an elliptic curve be $E : y^2 = x^3 + 5x + 4$ over $GF(13)$, $P = (0, 2)$ and $Q = (6, 4)$. Assume that we solve an integer scalar k such that $Q = [k]P$ by using *Baby-Step, Giant-Step method*. P has order 17. We first compute $m = |\sqrt{17}| = 5$. The points iP for $1 \leq i \leq 5$ are:

$$(0, 2), (4, 6), (10, 1), (6, 9), (8, 6).$$

We calculate $Q - jmP$ for $j = 0, 1, 2, 3, \ldots$ and obtain:

$$(6, 4), (11, 8), (10, 1), (4, 7), (1, 6)$$

at which point we stop since this third point matches $3P$. Since $j = 2$ yielded the match, we got:

$$(6, 4) = (3 + 2.5)P = 13P.$$

Therefore $k = 13$.

Binary Field. Let an elliptic curve be $E : y^2 + xy = x^3 + g^{11}x + g^{13}$ over GF(2^4), $P = (g^9, 1)$ and $Q = (g^6, g^6)$. Assume that we solve an integer scalar k such that $Q = [k]P$ by using *Baby-Step, Giant-Step method*. P has order 11. We first compute $m = \left\lceil \sqrt{11} \right\rceil = 4$. The points iP for $1 \leq i \leq 4$ are:

$(g^9, 1), (g^{12}, g^4), (g^6, 0), (g^{14}, 1)$.

We calculate $Q - jmP$ for $j = 0, 1, 2, 3, 4, \ldots$ and obtain:

$(g^6, g^6), (g^{14}, 1), O, (g^{14}, g^3), (g^6, 0)$.

at which point we stop since this second point matches $4P$. Since $j = 1$ yielded the match, we got:

$$(g^6, g^6) = ((4 + 1.4) \bmod 11)P = 8P.$$

Therefore $k = 8$.

5.2 Pollard's Rho Attack

Prime Field. Let an elliptic curve be $E : y^2 = x^3 + 5x + 4$ over $GF(13)$, $P = (0, 2)$ and $Q = (6, 4)$. Assume that we solve an integer scalar k such that $Q = [k]P$ by using *Pollard's rho method*. The point P has prime order 17. We choose $a, b \in [0, 17]$ uniformly at random, compute $R = [a]P + [b]Q$ and keep the triple (a, b, R) in the memory until we meet an another triple (a', b', R') such that $R = R'$ or $R = -R'$. Table (4) shows computing data used for Pollard's rho attack on $E : y^2 = x^3 + 5x + 4$

Table 4 Data for Pollard's rho attack on $E : y^2 = x^3 + 5x + 4$ over $GF(13)$

[a]	[b]	R = [a]P + [b]Q
5	**12**	**(11, 8)**
3	8	(8, 6)
10	4	(2, 3)
6	11	(6, 4)
2	**7**	**(11, 8)**
1	15	(11, 5)

Table 5 Data for Pollard's rho attack on $E : y^2 + xy = x^3 + g^{11}x + g^{13}$ over $GF(2^4)$

[a]	[b]	R = [a]P + [b]Q
10	**5**	$(\mathbf{g^{13}}, \mathbf{1})$
8	3	(g^9, g^7)
4	10	(g^{14}, g^3)
5	6	(g^{12}, g^6)
7	**4**	$(\mathbf{g^{13}}, \mathbf{1})$
2	7	$(g^6, 0)$

over $GF(13)$. We have that $[5]P + [12]Q = [2]P + [7]Q$. Then $k = (5 - 2)(7 - 12)^{-1} \bmod 17$; $k = 3(-5)^{-1} \bmod 17$; $k = 3.10 \bmod 17$; Hence $k = 13$.

Binary Field. Let an elliptic curve be $E : y^2 + xy = x^3 + g^{11}x + g^{13}$ over $GF(2^4)$, $P = (g^9, 1)$ and $Q = (g^6, g^6)$. Assume that we solve an integer scalar k such that $Q = [k]P$ by using *Pollard's rho method*. The point P has prime order 11. We choose $a, b \in [0, 11]$ uniformly at random, compute $R = [a]P + [b]Q$ and keep the triple (a, b, R) in the memory until we meet an another triple (a', b', R') such that $R = R'$ or $R = -R'$. Table (5) shows computing data used for Pollard's rho attack on $E : y^2 + xy = x^3 + g^{11}x + g^{13}$ over $GF(2^4)$. We have that $[10]P + [5]Q = [7]P + [4]Q$. Then $k = (10 - 7)(4 - 5)^{-1} \bmod 11$; $k = 3(-1)^{-1} \bmod 11$; $k = 3.10 \bmod 11$; Hence $k = 8$.

5.3 Pohlig–Hellman Attack

Prime Field. Let an elliptic curve be $E : y^2 = x^3 + 77x + 28$ over $GF(157)$, $P = (9, 115)$ and $Q = (2, 70)$. Assume that we solve an integer scalar k such that $Q = [k]P$ by using *Pohlig–Hellman method*. The order N of point P is 162. The prime factorization of N is 2.3^4. We compute k mod 2, and mod 81, then recombine them to obtain k mod 162 using the Chinese Remainder Theorem.

k mod 2. We compute $T = \{(24, 0)\}$.

Since $\frac{N}{2}.Q = (24, 0) = 1.(\frac{N}{2}.P)$, we have $k_0 = 1$.

Therefore $k \equiv 1 (\bmod 2)$.

k mod 81. We compute $T = \{(57, 41), (5, 99), (57, 116), O\}$.
Since $\frac{N}{3}.Q = (57, 41) = 1.(\frac{N}{3}.P)$, we have $k_0 = 1$.
Therefore $Q_1 = Q - 1.P = (5, 99)$.
Since $\frac{N}{9}.Q_1 = O = 0.(\frac{N}{3}.P)$, we have $k_1 = 0$.
Therefore $Q_2 = Q_1 - 0.3.P = Q_1$.
Since $\frac{N}{27}.Q_2 = (57, 116) = 2.(\frac{N}{3}.P)$, we have $k_2 = 2$.
Therefore $Q_3 = Q_2 - 2.9.P = (57, 41)$.
Since $\frac{N}{81}.Q_3 = (57, 116) = 2.(\frac{N}{3}.P)$, we have $k_3 = 2$.
Therefore $k = 1 + 0.3 + 2.9 + 2.27 \equiv 73 (\text{mod} 81)$.
We now have the simultaneous congruence:
$k \equiv 1$ (mod 2)
$k \equiv 73$ (mod 81).
Then we obtain $k = 73$ using the Chinese Remainder theorem to recombine simultaneous congruences as following:

$$M_1 = 162/2 = 81.$$
$$y_1 = M_1^{-1} \text{ mod } 2 = 1.$$
$$M_2 = 162/81 = 2.$$
$$y_2 = M_2^{-1} \text{ mod } 81 = 41.$$
$$k = 1.(81).1 + 73.(2).41(\text{ mod } 162) = 73.$$

Binary Field. Let an elliptic curve be $E : y^2 + xy = x^3 + g^{11}x + g^{13}$ over $GF(2^4)$, $P = (g^2, g^2)$ and $Q = (g^6, g^6)$. Assume that we solve an integer scalar k such that $Q = [k]P$ by using *Pohlig–Hellman method*. The order N of point P is 22. The prime factorization of N is 2.11. We compute k mod 2, and mod 11, then recombine them to obtain k mod 22 using the Chinese Remainder Theorem.

k mod 2. We compute $T = \{O\}$.
Since $\frac{N}{2}.Q = O = 0.(\frac{N}{2}.P)$, we have $k_0 = 0$.
Therefore $k \equiv 0 (\text{mod} 2)$.
k mod 11. We compute $T = \{(g^{13}, g^6)\}$.
Since $\frac{N}{11}.Q = (g^{13}, g^6) = 4.(\frac{N}{11}.P)$, we have $k_0 = 4$.
Therefore $k \equiv 4 (\text{mod} 11)$.
We now have the simultaneous congruence:
$k \equiv 0$ (mod 2)
$k \equiv 4$ (mod 11).
Then we obtain $k = 4$ using the Chinese Remainder theorem to recombine simultaneous congruences as following:

Table 6 Time complexity

Attacks	Expected running time
Baby-Step Giant-Step	$O(\sqrt{N})$
Pollard's rho	$O(\sqrt{\pi N}/2)$
Pohlig–Hellman	$O(\sqrt{q})$

$$M_1 = 22/2 = 11.$$
$$y_1 = M_1^{-1} \bmod 2 = 1.$$
$$M_2 = 22/11 = 2.$$
$$y_2 = M_2^{-1} \bmod 11 = 6.$$
$$k = 0.(11).1 + 4.(2).6(\bmod 22) = 4.$$

6 Conclusion

The security strong point of ECC relies on the complexity of solving ECDLP for a cryptanalyst to find the secret key k such that $Q = kP$. The Table (6) summarizes time complexity of general methods of attacking on ECDLP. Our research found that these attacking methods can solve ECDLP within the corresponding expected running time when the group order N of the elliptic curve is not enough large and its prime factorization is composed of smooth primes.

When implementing the ECC, the following several classes of elliptic curves should be applied in order to gain the maximum security level of the cryptosystems. The National Institute of Standards and Technology (NIST) issued several classic elliptic curves with larger key sizes for federal government use.

NIST recommends the 15 elliptic curves: five elliptic curves over $GF(p)$ where p equals 192, 224, 256, 384, and 521 bits and five elliptic curves over $GF(2^m)$ where m equals 163, 233, 283, 409, and 571. For each of the binary fields, one Koblitz curve is recommended [12]. Thus, NIST issue contains a total of five prime curves and ten binary curves. These curves should be selected for best security and implementation efficiency. The group order for each of these curves is enough large and has large prime factors. Therefore, these curves are resistant to the attacking methods we studied in the Sect. 4.

References

1. Anoop, M.S.: Elliptic Curve Cryptography. http://www.infosecwriters.com/Papers/Anoopms_ECC.pdf
2. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer press (2004)
3. Lidl, R., Niederreiter, H.: Introduction to Finite Field Arithmetic and their Applications. Cambridge University Press (1986)
4. Behrouz, A.: Forouzan, Cryptography and Network Security. McGraw-Hill press, International Edition (2008)
5. Liao, H.-Z., Shen, Y.-Y.: On the elliptic curve digital signature algorithm. Tunghai Sci. **8** (2006)
6. Karthikeyan, E.: Survey of elliptic curve scalar multiplication algorithms. Int. J. Adv. Netw. Appl. **04**(02) (2012)
7. Washington, L.C.: Elliptic Curves: Number Theory and Cryptography. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL (2003)
8. Musson, M.: Attacking the elliptic curve discrete logarithm problem. Master Thesis of Science (Mathematics and Statistics) Acadia University (2006)
9. Rosen, K.H.: Discrete Mathematics and its Applications, Global Edition (2008)
10. Hla, N.N., Aung, T.M.: Implementation of finite field arithmetic operations for large prime and binary fields using java BigInteger class. Int. J. Eng. Res. Technol. (IJERT), **6**(08) (2017)
11. Aung, T.M., Hla, N.N.: Implementation of elliptic curve arithmetic operations for prime field and binary field using java BigInteger class. Int. J. Eng. Res. Technol. (IJERT), **6**(08) (2017)
12. Recommended Elliptic Curves for Federal Government Use, NIST (1999)