International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:11, No:11, 2017

A Study of General Attacks on Elliptic Curve Discrete Logarithm Problem over Prime Field and Binary Field

Tun Myat Aung, Ni Ni Hla

***Abstract*—**This paper begins by describing basic properties of finite field and elliptic curve cryptography over prime field and binary field. Then we discuss the discrete logarithm problem for elliptic curves and its properties. We study the general common attacks on elliptic curve discrete logarithm problem such as the Baby Step, Giant Step method, Pollard's rho method and Pohlig-Hellman method, and describe in detail experiments of these attacks over prime field and binary field. The paper finishes by describing expected running time of the attacks and suggesting strong elliptic curves that are not susceptible to these attacks.

***Keywords*—**Discrete logarithm problem, general attacks, elliptic curves, strong curves, prime field, binary field, attack experiments.

I. Introduction

ELLIPTIC Curve Cryptography (ECC) is an alternative approach for implementing public-key cryptography (PKC) in which each entity (user or device) taking part in the communication 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 particular entity keeps the private key in secret but the public key is distributed to all entities taking part in the communication [1]. ECC can be used for providing the following security services:

- confidentiality,
- authentication,
- data integrity,
- non-repudiation,
- authenticated key exchange.

Nowadays, ECC becomes a leader in the industry of information security technology. It replaces other public key cryptosystems such as RSA and DSA. It becomes the industrial standard. This is a result of an increase in speed and lower power consumption during implementation due to less memory usage and smaller key sizes. Its security depends on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP). Although the ECDLP is thought to be an intractable problem, it has not stopped attackers/intruders attempting to attack on elliptic curve cryptosystems. Various attacks have been invented, tested and analyzed by many mathematicians over the years, in efforts to find flaws in elliptic curve cryptosystems. Some attacks have been partially successful, but others have not.

The purpose of this paper is to study the general common attacks against the ECDLP and to apply the knowledge of them in an effort to choose cryptographically strong elliptic curves over prime field and binary field under large integer. The organization of this paper is as follows. Section II includes finite field and its properties. In Section III, we discuss ECC over prime field and binary field and its geometric properties. Section IV describes in details the ECDLP, its properties and its general common attacks. In Section V, we discuss our attack experiments over prime field and binary field. Finally, in Section VI we conclude our discussion by describing expected running time of the attacks and by suggesting strong curves for secure implementation of ECC systems.

Tun Myat Aung and Ni Ni Hla are with the University of Computer Studies, Yangon (UCSY), Myanmar (e-mail: tma.mephi@gmail.com).

II. Finite Field Arithmetic

A finite field, denoted by F, is a field containing a finite number of elements. *Fields* are used to number systems such as the rational numbers, the real numbers, and the complex numbers. They consist of a set of elements that can perform two arithmetic operations: *addition* denoted by (+) and *multiplication* denoted by (·). They satisfy the following arithmetic properties:

- (F,+) is a finite group with *additive* identity denoted by 0.
- (F\{0}, ·) is a finite group with *multiplicative* identity denoted by 1.
- Elements of finite group follow the distributive law: (a+b) · c = (a · c) + (b · c) for all a, b, c ∈ F.

If the elements of the field are finite, then the field is said to be *finite* [3]. Galois presented 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 usually called *Galois fields* and also denoted as $GF(p^m)$. If $m = 1$, then the field $GF(p)$ is called a *prime field*. If $m \geq 2$, then the field $GF(p^m)$ is called an *extension field*. The number of elements in a finite field is the *order* of the field. Any two fields are *isomorphic* if their orders are the same [11].

A. Field Operations

A finite field F has two arithmetic operations, *addition* and *multiplication*. However, the s*ubtraction* of elements in a finite field is defined in the expression of addition. For instance, let $a, b \in F$, $a - b$ is defined as $a + (-b)$, in this case $-b$ is the single element in the field such that $b + (-b) = 0$. $-b$ is called *additive inverse* of b. Similarly, the *division* of elements in a finite field is defined in the expression of multiplication. For instance, let $a, b \in F$ with $b \neq 0$, a/b is defined as $a \cdot b^{-1}$, in this case b^{-1} is the single element in the field such that $b \cdot b^{-1} = 1$ [3]. b^{-1} is called the *multiplicative inverse* of b.

B. Prime Field

Let p be a prime number. A set of integer elements modulo p, consisting of the integers $\{0,1,2,\ldots, p-1\}$ with addition and multiplication performed modulo p, is a finite field of prime order p. It is called *prime field* denoted by *GF(p)* and p is called the *modulus* of *GF(p)*. For any integer a, $a \bmod p$ denotes the integer remainder r obtained upon dividing a by p. This operation is called *reduction modulo p*. The remainder r is the single integer element between 0 and $p-1$, i.e. $0 \leq r \leq p-1$ [3].

Example 1. (*prime field* GF(29)) The elements of GF(29) are $\{0,1,2,\ldots,28\}$. The following are some examples of arithmetic operations in GF(29).

(a). Addition: $27+10 = 8$ since $37 \bmod 29 = 8$.
(b). Subtraction: $10-27 = 12$ since $-17 \bmod 29 = 12$.
(c). Multiplication: $27 \cdot 10 = 9$ since $270 \bmod 29 = 9$.
(d). Inversion: $27^{-1} = 14$ since $27 \cdot 14 \bmod 29 = 1$.

C. Binary Field

A finite field of order 2^m is called *binary field* denoted by *GF(2^m)*. It also refers to the *finite field with characteristic-two*. One approach to construct *GF(2^m)* is to apply a *polynomial basis representation* denoted by (1). In this case, the elements of *GF(2^m)* are the binary polynomials of degree at most $m-1$.

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

An irreducible binary polynomial *f (x)* of degree m is chosen. Irreducibility of *f(x)* means that *f(x)* cannot be factored as a product of binary polynomials each of degree less than m. Addition of binary field elements is the usual addition of polynomials, with coefficient arithmetic performed modulo 2. Multiplication of binary field elements is performed modulo the *reduction polynomial f(x)*. For any binary polynomial *a(x)*, *a(x)* mod *f(x)* shall denote the unique remainder polynomial *r(x)* of degree less than m obtained upon long division of *a(x)* by *f(x)*; this operation is called *reduction modulo f(x)* [3].

Example 2. (*Binary Field GF(2^4)*). In Table I, the elements of *GF(2^4)* are the 16 binary polynomials of degree at most 3.

TABLE I
BINARY POLYNOMIALS

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 following are some examples of arithmetic operations in *GF(2^4)* with reduction Polynomial $f(x) = x^4 + x + 1$.

(a). Addition: $(x^3+x^2+1)+(x^2+x+1) = x^3+x$.
(b). Subtraction: $(x^3+x^2+1)-(x^2+x+1) = x^3+x$.
(c). Multiplication: $(x^3+x^2+1).(x^2+x+1) = x^2+1$ since $(x^3+x^2+1).(x^2+x+1) = x^5+x+1$ and $(x^5+x+1) \bmod (x^4+x+1) = x^2+1$.
(d). Inversion: $(x^3+x^2+1)^{-1} = x^2$ since $(x^3+x^2+1).x^2 \bmod (x^4+x+1) = 1$.

III. Elliptic Curve Arithmetic

A. Elliptic Curves over Prime Field -GF(p)

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. The following elliptic curves are adopted from the general Weierstrass equation. The elliptic curve E(GF(p)) over prime field GF(p) is defined by (2) [2]:

$$y^2 = x^3 + ax + b \quad (2)$$

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

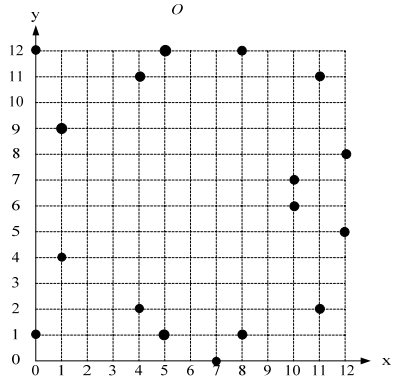
1).Points on E(GF(p))

The elliptic curve E(GF(p)) consists of a set of points $\{P = (x,y) | y^2 = x^3 + ax + b, x, y, a, b \in GF(p)\}$ together with a point at infinity denoted as *O*. Every point on the curve has its *inverse*. The inverse of a point (x, y) on E(GF(p)) is (x, $-y$). The number of points on the curve, including a point at infinity, is called its *order #E*. The pseudocode for finding the points on the elliptic curve E(GF(p)) is shown in Algorithm (1).

Algorithm (1). Pseudocode for finding the points on the elliptic curve 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

(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

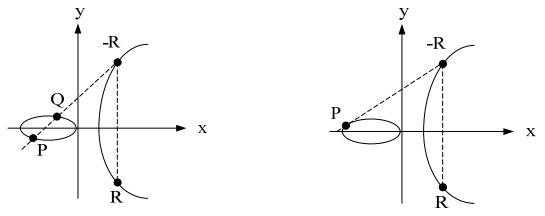
Fig. 1 Points on $E: y^2 = x^3 + 5x + 4$

Example 3. Let p = 13 and consider the elliptic curve $E: y^2 = x^3 + 5x + 4$ defined over GF(p) 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 Figs. 1 (a) and (b). The *order* of the elliptic curve $E: y^2 = x^3 + 5x + 4$ over GF(13) is 17.

2).Arithmetic Operations on E(GF(p))

The *chord-and-tangent rule* is applied for adding two points on an elliptic curve E(GF(p)) to give a third point on the curve. Using this 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 that is used in the construction of elliptic curve cryptosystems [5]. The addition rule is the best to explain geometrically. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two distinct points on an elliptic curve E(GF(p)). Then the third point $R = (x_3, y_3)$ is obtained by *addition* of *P* and *Q* as follows. First draw the line through *P* and *Q*; this line intersects the elliptic curve in a third point. Then *R* is the reflection of this point in the *x*-axis. This is illustrated in Fig. 2 (a). The elliptic curve in the figure consists of two parts, the ellipse-like figure and the infinite curve. If $P = (x_1, y_1)$, then the *double* of *P*, denoted $R = (x_3, y_3)$, is defined as follows. First draw the tangent line to the elliptic curve at *P*. This line intersects the elliptic curve in a second point. Then *R* is the reflection of this point in the *x*-axis. This is illustrated in Fig. 2 (b).



(a) Addition ($R = P + Q$) (b) Doubling ($R = P + P$)

Fig. 2 Geometric Description

The following algebraic methods for the addition of two points and the doubling of a point can be resulted from the geometric description [2].

(a). $P + O = O + P = P$ for all $P \in E(GF(p))$.

(b). If $P = (x, y) \in E(GF(p))$, then $(x, y) + (x, -y) = O$. The point (*x*, -*y*) denoted by (-*P*) is called the *inverse* of *P*; –*P* is a point on the curve.

(c). (*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, $\lambda = (y_2 - y_1)/(x_2 - x_1)$.

(d). (*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, $\lambda = (3{x_1}^2 + a)/2y_1$.

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

a. *Addition*. Let $P = (1, 6)$ and $Q = (4, 6)$. Then $P + Q = (8, 7)$.

b. *Doubling*. Let $P = (1, 6)$. Then $2P = (10, 1)$.

c. *Inverse*. Let $P = (1, 6)$. Then $-P = (1, 7)$.

B. Elliptic Curves over Binary Field - GF(2^m)

A reduction polynomial $f(x)$ must be firstly chosen to construct a binary field GF(2^m). The elements generated by the reduction polynomial are applied to construct an elliptic curve E(GF(2^m)). The elliptic curve E(GF(2^m)) over binary field GF(2^m) is defined by (3) [2]:

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

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

1).Points on E(GF(2^m))

The elliptic curve E(GF(2^m)) consists of a set of points: $\{P = (x, y) | y^2 + xy = x^3 + ax + b, x, y, a, b \in GF(2^m)\}$ together with a point at infinity denoted as *O*. Every point on the curve has its *inverse*. The inverse of a point (*x*, *y*) on E(GF(2^m)) is $(x, x \oplus y)$. The number of points on the curve, including a point at infinity, is called its *order* #*E*. The pseudocode for finding the points on the elliptic curve E(GF(2^m)) is shown in Algorithm (2).

Algorithm (2). Pseudocode for finding the points on the elliptic curve E(GF(2^m))

Input: a, b, $f(x)$
Output: $P_i = (x_i, y_i)$
Begin
$x_i = \{0, 1, g^1, \ldots, g^{m-2}\}$
$y_j = \{0, 1, g^1, \ldots, g^{m-2}\}$

for(i=0; i<2^m*; i++){*
  for(j=0; j < 2^m*;j++){*

  $w_1 = {x_i}^3 \oplus a x_i \oplus b.$
  $w_2 = {y_j}^2 \oplus x_i y_j$
  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 16 elements of GF(2^4) are shown in Table II.

TABLE II
ELEMENTS OF GF(2^4)

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

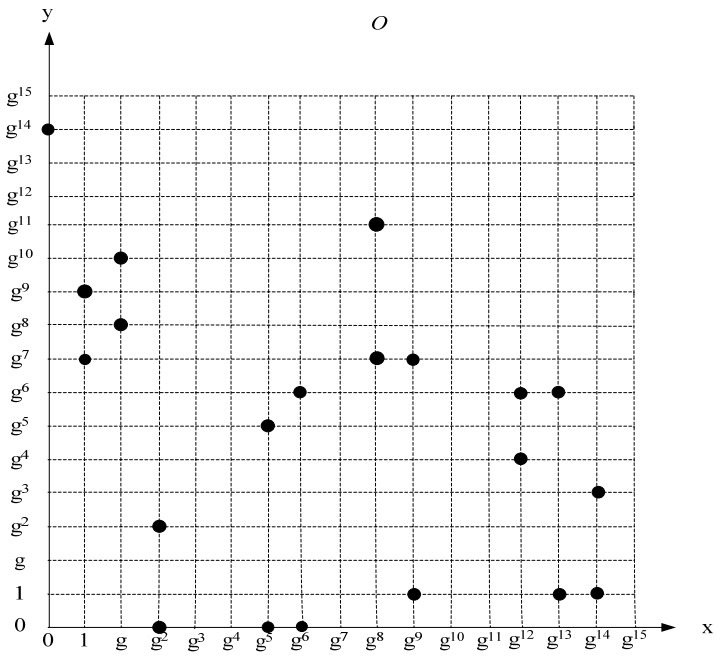
Table III shows the power representation of g for elements of GF(2^4) generated by the polynomial $f(x) = x^4 + x + 1$. The element of $g = x = (0010)$ is a generator of GF(2^4) because its order is 15 ($2^4 - 1$).

Using the elliptic curve $E: y^2 + xy = x^3 + g^{11}x + g^{13}$, with $a = g^{11}$ and $b = g^{13}$, we can find the points on the curve, as

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

$(0, g^{14})$	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 $E: y^2 + xy = x^3 + g^{11}x + g^{13}$

TABLE III
POWER REPRESENTATION OF ELEMENTS

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		

2).Arithmetic Operations on E(GF(2^m))

As with elliptic curves over GF(p), the *chord-and-tangent rule* is also applied for adding two points on an elliptic curve E(GF(2^m)) to give a third point on the curve. Using this addition operation with points on E(GF(2^m)) generates a group with O serving as its identity [5]. The algebraic methods for the addition of two points and the doubling of a point are the following [2].

(a). $P + O = O + P = P$ for all $P \in E(GF(2^m))$.

(b). If $P = (x, y) \in E(GF(2^m))$, then $(x, y) + (x, x + y) = O$. The point $(x, x+y)$ denoted by $(-P)$ is called the *inverse* of P; $-P$ is a point on the curve.

(c). (*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, $\lambda = (y_2 + y_1)/(x_2 + x_1)$.

(d). (*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.

a. *Addition*. Let $P = (g^2, g^2)$ and $Q = (g^6, g^6)$. Then $P + Q = (g^5, 0)$.

b. *Doubling*. Let $P = (g^2, g^2)$. Then $2P = (g^{14}, 1)$.

c. *Inverse*. Let $P = (g^2, g^2)$. Then $-P = (g^2, 0)$.

III. ELLIPTIC CURVE DISCRETE LOGARITHM PROBLEM

The security of ECC depends on the ability to solve ECDLP. Let P be a point on an elliptic curve and point Q is a point on the curve such that $Q = kP$, where k is an integer. Given two points, P and Q, it is not able to compute k, if the group order of the points is sufficiently large. k is called the discrete logarithm of Q to the base P.

A. Point Multiplication

Another main operation involved in ECC is *point multiplication*. The multiplication of a scalar k with any point P on the curve generates another point Q on the curve [1]. This is achieved by repeating *point addition and doubling* operations based on binary representation of integer k. The binary representation of integer k is shown as (4)

$$k = k_{n-1}2^{n-1} + k_{n-2}2^{n-2} + \cdots + k_1 + k_0 \qquad (4)$$

where $k_{n-1} = 1$ and $k_i \in \{0, 1\}, i = 0, 1, 2, \ldots, n-1$. This method is called *binary method* [3] which scans the bits of k either from left-to-right or right-to-left. Algorithm 3 given illustrates the computation of kP using binary method. It can be used for both elliptic curves over prime field GF(p) and binary field GF(2^m).

Algorthm (3). Binary Method

Input: point P and binary representation of integer k
Output: point Q such that Q = kP
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

The cost of multiplication depends on the number of 1s in *binary representation* of *k*. The number of 1s is called the *Hamming Weight* of scalar. In an average, binary method requires (n-1) *point doublings* and (n-1)/2 *point additions*. For each bit .1., we need to perform *point doubling* and *point addition*, if the bit is .0., we need only *point doubling* operation. Therefore, reducing the number of 1s in the binary representation will improve the speed of elliptic curve scalar multiplication [4].

B. Order of Points

Let $P \in E$(GF(p)). The *order* of P is the smallest positive integer, N, such that $NP = O$ where O is the group identity. Hasse's theorem proved (5) [7].

$$p + 1 - 2\sqrt{p} \leq N \leq p + 1 + 2\sqrt{p}. \quad (5)$$

All values of N need to be tried in this range and see which ones satisfy $NP = O$.

Example 7. Let E be the elliptic curve $E: y^2 = x^3 + 5x + 4$ over GF(13). The point (0, 2) can be shown to have order 17. Hasse's theorem says that $13 + 1 - 2\sqrt{13} \leq N \leq 13 + 1 + 2\sqrt{13}$; we could try all values of N in this range, $7 \leq N \leq 21$, and find 17 that satisfy $NP = O$. Therefore, N = 17.

C. Attacks on ECDLP

The discrete logarithm problem is fundamentally important to the area of PKC. Almost all of the most commonly used public key cryptographic systems are based on the assumption that the discrete logarithm is extremely difficult to compute; the more difficult it is, the more security it supports. One way to increase the difficulty of the discrete logarithm problem is to base the public key cryptosystems on a larger group order under large integer.

The following algorithms can solve the elliptic curve discrete logarithm under small integer. General attacks on the ECDLP can be grouped into three classes [8]:

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

We studied the following general common attacks on the ECDLP.

1).Baby-Step Giant-Step Method

This method was developed by D. Shanks for computations in algebraic number theory. Let $P,Q \in E$(GF(p)). Suppose that we want to solve $Q = [k]P$. P has prime order N. First, we need to find the order N of P. The method requires approximately $\sqrt{N}$ steps and around $\sqrt{N}$ storage. Therefore it only works well for moderate sized N. The procedure is as follows [7].

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

The points for iP are calculated by adding P (a "*baby step*") to $(i - 1)P$. The points for $Q - jmP$ are computed by adding $-mP$ (a "*giant step*") to $Q - (j - 1)mP$. The algorithm as presented above may require roughly m steps to find a match and expected running time is $O(\sqrt{N})$ [7].

2).Pollard's Rho Method

Let E(GF(p)) be an elliptic curve and $P \in E(GF(p))$. Suppose that P has order N, where N is prime, and let $Q \in < P >$. Suppose that we want to solve $Q = [k]P$. In this attack we will attempt to find distinct pairs of integers (a, b) and (a', b') modulo N such that $[a]P + [b]Q = [a']P + [b']Q$. One method for finding these pairs of integers is to simply select $a, b \in [0, N - 1]$ uniformly at random, compute the point $[a]P + [b]Q$, and then store the triple $(a, b, [a]P + [b]Q)$. We continue to generate pairs (a, b) uniformly at random and check these against all previously stored triples until we find a pair (a', b') with $[a']P + [b']Q$ where $(a, b) \neq (a', b')$. When this happens we have solved the ECDLP and as mentioned above, we can rearrange (6)

$$[a]P + [b]Q = [a']P + [b']Q \quad (6)$$

as $[a - a']P = [b' - b]Q = [b' - b]([k]P)$, and thus $k \equiv (a - a')(b' - b)^{-1} mod\ N$. This first method gives an expected running time of $O(\sqrt{\pi N}/2)$ [7], but unfortunately requires approximately $O(\sqrt{\pi N}/2)$ amount of storage for the triples that we have computed.

A second approach that has roughly the same running time, but uses less storage is also known. Instead of storing a list of triples, we define a function $f: < P > \rightarrow < P >$ so that for any $R \in < P >$ and $a, b \in [0, N - 1]$ with $R = [a]P + [b]Q$, we can easily compute $R = f(R')$ and $a', b' \in [0, N - 1]$ with $R' = [a']P + [b']Q$. One way to define such a function is to partition $< P >$ into L sets of roughly equal size, say $\{S_1, S_2, \ldots, S_L\}$. We define a second function H so that $H(X) = j$ if $X \in S_j$. Then $a_j, b_j \in [0, N - 1]$ are chosen uniformly at random for $1 \leq j \leq L$. Now our function $f: < P > \rightarrow < P >$ is defined by (7)

$$f(R) = R + [a_j]P + [b_j]Q, where\ j = H(X). \quad (7)$$

So, if $R = [a]P + [b]Q$, then $f(R) = R' = [a']P + [b']Q$ where $a' = a + a_j\ mod\ n$ and $b' = b + b_j\ mod\ N$. This then determines a sequence of points in $< P >$. Since $< P >$ is finite we will eventually obtain a collision, thus obtaining our pairs of

integers (a, b) and (a', b'), and so enabling us to solve the ECDLP. As mentioned, this approach has a similar running time to the first, but requires less storage, since we are no longer required to store ordered triples in order to find a collision. The diagram of the sequence looks like the Greek letter ρ. That is why this method is called the Pollard-Rho method.

3).Pohlig-Hellman Method

This method is a special purpose algorithm used for computing discrete logarithms in a multiplicative group with order of smooth integer.

Let P and Q be points on an elliptic curve. Suppose that we want to solve an integer k such that $Q = [k]P$. In this attack we know the order N of P and we first compute the prime factorization of N satisfied by (8):

$$N = \prod_i {q_i}^{e_i} \tag{8}$$

The idea of this algorithm is to find $k \pmod{{q_i}^{e_i}}$ for each i, use the Chinese Remainder theorem [6] to combine them and then obtain $k \pmod N$. Let q be a prime, and let q^e be the exact power of q dividing N. Write k in its base q expansion as (9)

$$k = k_0 + k_1 q + k_2 q^2 + \cdots \tag{9}$$

where $0 \le k_i < q$. We will evaluate $k \,(mod\, q^e)$ by successively determining $k_0, k_1, k_2, \ldots, k_{e-1}$. The procedure is as follows [7]:

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

Then $k \equiv k_0 + k_1 q + \cdots + k_{e-1} q_{e-1} \,(mod\, q_e)$. Therefore we find k_1. Similarly, the method produces k_2, k_3, ...We have to stop after r = e-1. The expected running time of this algorithm is $O(\sqrt{q})$ [7], where q is the largest prime divisor of N. In practice this attack becomes infeasible when N has a large prime divisor. If this is the case, it then becomes difficult to make and store the list T to find matches.

V. Experiments

We implemented well-known general common attacks such as Baby-Step Giant-Step algorithm, Pollard's rho method and the Pohlig-Hellman method by using our implementations of finite field arithmetic operations [9] and elliptic curve arithmetic operations [12] under java BigInteger class.

A. 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). We suppose that we determine the unique integer k such that $Q = [k]P$ by using *Baby-Step, Giant-Step method*. P has order 17. We first compute $m = \lceil\sqrt{17}\rceil = 4$. The points iP for $1 \le i \le 4$ are **(0, 2)**, (4, 6), (10, 1), (6, 9). We calculate $Q - jmP$ for j = 0, 1, 2, 3, … and obtain (6, 4), (11, 5), (8, 6), **(0, 2)** at which point we stop since this fourth point matches P. Since j = 3 yielded the match, we have (6, 4) = $(1 + 3 \cdot 4)P = 13P$.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 $Q = (g^6, g^6)$. We suppose that we determine the unique integer k such that $Q = [k]P$ by using *Baby-Step, Giant-Step method*. P has order 11. We first compute $m = \lceil\sqrt{11}\rceil = 4$. The points iP for $1 \le i \le 4$ are $(g^9, 1)$, (g^{12}, g^4), $(\boldsymbol{g^6}, \mathbf{0})$, $(g^{14}, 1)$. We calculate $Q - jmP$ for j = 0, 1, 2, 3, 4, … and obtain (g^6, g^6), $(g^{14}, 1)$, O, (g^{14}, g^3), $(\boldsymbol{g^6}, \mathbf{0})$, at which point we stop since this fourth point matches 3P. Since j = 4 yielded the match, we have (g^6, g^6) = ((3 + 4 · 4) mod 11) P = 8P. Therefore k = 8.

B. 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). We suppose that we determine the unique integer k such that $Q = [k]P$ by using *Pollard's rho method*. The base point P has prime order 17. We choose uniformly at random $a, b \in [0, 17]$, calculate $R = [a]P + [b]Q$ and store the triple (a, b, R) until such time we encounter a second triple (a', b', R') such that $R = R'$ or $R = -R'$. We have that [5]P + [12]Q = [2]P + [7]Q. Then $k = (5 - 2)(7 - 12)^{-1} mod\ 17$; $k = 3(-5)^{-1} mod\ 17$; $k = 3.10\ mod\ 17$; hence k =13.

TABLE IV
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)
7	10	(0,2)
⋮	⋮	⋮

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 $Q = (g^6, g^6)$. We suppose that we determine the unique integer k such that $Q = [k]P$ by using *Pollard's rho method*. The base point P has prime order 11. We choose uniformly at random $a, b \in [0, 11]$, calculate $R = [a]P + [b]Q$ and store the triple (a, b, R) until such time we encounter a second triple (a', b', R') such that $R = R'$ or $R = -R'$. We have that [10]P + [5]Q = [7]P + [4]Q. Then $k = (10 - 7)(4 - 5)^{-1} mod\ 11$; $k = 3(-1)^{-1} mod\ 11$; $k = 3.10\ mod\ 11$; hence k =8.

TABLE V
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**	$(\boldsymbol{g^{13}}, \mathbf{1})$
8	3	(g^9, g^7)
4	10	(g^{14}, g^3)
5	6	(g^{12}, g^6)
7	**4**	$(\boldsymbol{g^{13}}, \mathbf{1})$
2	7	$(g^6, 0)$
⋮	⋮	⋮

C. 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). We suppose that we determine the unique integer 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 $\mathbf{2.3^4}$. We'll 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 $\boldsymbol{k \equiv 1\ (mod\ 2)}$.

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 $\boldsymbol{k = 1 + 0.3 + 2.9 + 2.27 \equiv 73\ (mod\ 81)}$.

We now have the simultaneous congruence:

$$k \equiv 1 \pmod 2$$
$$k \equiv 73 \pmod{81}.$$

Then we use the Chinese Remainder theorem to recombine these, and we obtain $k = 73$. $M_1 = 162/2 = 81$. $y_1 = M_1^{-1}\ mod\ 2 = 1$. $M_2 = 162/81 = 2$. $y_2 = M_2^{-1}\ mod\ 81 = 41$. $k = 1.(81).1 + 73.(2).41\ (mod\ 162) = 73$.

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 $Q = (g^6, g^6)$. We suppose that we determine the unique integer 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 $\mathbf{2.11}$. We'll 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 $\boldsymbol{k \equiv 0\ (mod\ 2)}$.

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 $\boldsymbol{k \equiv 4\ (mod\ 11)}$.

We now have the simultaneous congruence:

$$k \equiv 0 \pmod 2$$
$$k \equiv 4 \pmod{11}.$$

Then we use the Chinese Remainder theorem to recombine these, and we obtain $k = 4$. $M_1 = 22/2 = 11$. $y_1 = M_1^{-1}\ mod\ 2 = 1$. $M_2 = 22/11 = 2$. $y_2 = M_2^{-1}\ mod\ 11 = 6$. $k = 0.(11).1 + 4.(2).6\ (mod\ 22) = 4$.

VI. CONCLUSION

The cryptographic strength of elliptic curve cryptosystems lies in the difficulty of solving ECDLP for a cryptanalyst to determine the secret random number k from kP. Table VI summarizes the expected running time of our general common attacks. Our research study found that these general common attacks can solve ECDLP within the following corresponding expected running time when the group order N of the elliptic curve is small and its prime factorization is composed of small primes. All of the general common attacks on the ECDLP are expected to run in fully exponential time.

TABLE VI
EXPECTED RUNNING TIME

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

When implementing the elliptic curve cryptosystem, the following several classes of elliptic curves should be used if we want to achieve the maximum possible security level of the cryptosystems. T The National Institute of Standards and Technology (NIST) submitted a report to recommend a set of elliptic curves with larger key sizes for federal government use [10].

NIST recommends the following fifteen elliptic curves.

- Five elliptic curves over prime fields GF(p) for certain primes p of sizes 192, 224, 256, 384, and 521 bits. [10].
- Five elliptic curves over binary fields GF(2^m) for m equal 163, 233, 283, 409, and 571. For each of the binary fields, one Koblitz curve is recommended [10].

The NIST recommendation thus contains a total of five prime curves and ten binary curves. These curves should be chosen for optimal security and implementation efficiency. The group order for each of these curves is large and has large prime factors. Therefore, these curves are resistant to the attacks described above.

REFERENCES

[1] Anoop MS. Elliptic Curve Cryptography, http://www.infosecwriters.com/Papers/Anoopms_ECC.pdf.
[2] Behrouz A. Forouzan, Cryptography and Network Security, McGraw-Hill press, International Edition, 2008.
[3] Darrel Hankerson, Alfred Menezes, Scott Vanstone. Guide to Elliptic Curve Cryptography, Springer press, 2004.
[4] E. Karthikeyan. Survey of Elliptic Curve Scalar Multiplication Algorithms, Int. J. Advanced Networking and Applications, Volume 04, Issue 02, 2012.
[5] Hung-Zih Liao, Yuan-Yuan Shen. On the Elliptic Curve Digital Signature Algorithm, Tunghai Science Volume 8, July, 2006.
[6] Kenneth H. Rosen. Discrete Mathematics and its Applications, Global Edition, 2008.
[7] Lawrence C. Washington. Elliptic curves. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2003. Number theory and cryptography.

[8] Matthew Musson. Attacking the Elliptic Curve Discrete Logarithm Problem, Master Thesis of Science (Mathematics and Statistics) Acadia University, 2006.
[9] Ni Ni Hla, Tun Myat Aung. Implementation of Finite Field Arithmetic Operations for Large Prime and Binary Fields using java BigInteger class, International Journal of Engineering Research and Technology (IJERT), Volume 6, Issue 08, August – 2017
[10] Recommended Elliptic Curves for Federal Government Use, NIST, 1999.
[11] Rudolf Lidl and Harald Niederreiter, Introduction to Finite Field Arithmetic and their Applications, Cambridge University Press, 1986.
[12] Tun Myat Aung, Ni Ni Hla. Implementation of Elliptic Curve Arithmetic Operations for Prime Field and Binary Field using java BigInteger class, International Journal of Engineering Research and Technology (IJERT), Volume 6, Issue 08, August - 2017.

Tun Myat Aung passed matriculation exam with 3 subject distinctions and got B.Econ from Institute of Economics, Yangon, and M.I.Sc. from University of Computer Studies, Yangon (UCSY), and M.Engnn & Tech. (I.T) and Ph.D (IT) from National Research Nuclear University M.E.Ph.I (Moscow). He is a professor in University of Computer Studies, Yangon (UCSY). His main research interests include programming languages, distributed and mobile systems, computer security and privacy, software vulnerabilities and security, e-commerce, cryptography, stenography, network and information security, and symbolic computation.

Ni Ni Hla holds M.Sc (mathematics) and M.I.Sc, and is a lecturer in University of Computer Studies, Yangon (UCSY). Her main research interests include cryptography, network and information security, symbolic computation and mathematics of computing.