A NEW PROPOSED METHOD FOR SOLVING AN ELLIPTIC CURVE DISCRETE LOGARITHM PROBLEM

Ammar Ali Neamah Alrammahi
Mathematical Department,
Collage of Mathematics and Computer Science
Kufa University

Abstract

In this work, we present a new approach for solving Elliptic Curve Discrete Logarithm Problem. This method provides a new access to the field of attacking methods the Elliptic Curve Cryptosystems. Beside this paper gives a program for executing the proposed method by using MATLAP.

1.Introduction //

Many papers were published about elliptic curve and the ECDLP related to them. For good survey one can turn to [3]. An Elliptic Curve Cryptosystems (ECC_s) works in a cyclic subgroup of order n of the group formed by an elliptic curve (for more details, see Koblitz [4] or Menezes [5]). A user selects a generator P of a cyclic subgroup $\langle P \rangle = \{\mathbf{O}_\infty, P, [2]P, \ldots, [n-1]P\}$ and randomly generates his private key k between 1 and n – 1. These variables are used to calculate the public key Q = [k]P. Everything except k is publicly known. The cryptosystem can be broken by solving the ECDLP that is calculating k knowing only P, Q, denoted $\log_P Q$ [1]. The security of these cryptosystems relies on the difficulty of solving the ECDLP. If this problem can be solved efficiently, then elliptic curve based cryptosystems can be broken efficiently [6].

2. Background

This section will provide the necessary background material on various properties of elliptic curves and will also describe the elliptic curve discrete logarithm problem.

2.1 Elliptic Curves Over F_p //

Let F_p be a finite field of characteristic $p \neq 2, 3$, and let $a, b \in F_p$ satisfy the inequality $4a^3 + 27b^2 \neq 0$. An elliptic curve **E** defined over field F_p is defined as the set of points $(x, y) \in F_p \times F_p$ which satisfy the equation

$$y^2 = x^3 + ax + b, \quad (1)$$

together with a special point, $\mathbf{O}_\infty$, called the point at infinity.
i.e. $\mathbf{E}(F_p) = \{(x, y) : x, y \in F_p : y^2 = x^3 + ax + b\} \cup \{\mathbf{O}_\infty\}$. These points form an abelian group under a well-defined addition operation which we now describe.

Let $\mathbf{E}(F_p)$ be an elliptic curve and let P and Q be two points on $\mathbf{E}(F_p)$. If $P = \mathbf{O}_\infty$, then $\ominus P = \mathbf{O}_\infty$ and $P \oplus Q = Q \oplus P = Q$. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$. Then $\ominus P = (x_1, -y_1)$ and $P \oplus (\ominus P) = \mathbf{O}_\infty$. If $Q \neq \ominus P$ then $P \oplus Q = (x_3, y_3)$ where [7]:

$$x_3 = \lambda^2 - x_1 - x_2 \quad (2)$$
$$y_3 = \lambda \cdot (x_1 - x_3) - y_1 \quad (3)$$

and

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1} & \text{if } P = Q \end{cases} \qquad (4)$$

2.2 Elliptic Curve Discrete Logarithm Problem //

A necessary condition for the security of all ECC_s is that the ECDLP be intractable. In this problem, one is given an elliptic curve **E** defined over a finite field F_p, a point P of order n on **E**, and a point Q that is a multiple of P, and one has to find the integer k such that Q = [k]P. In ECC_s, the non-secret parameters E , a , b , F_p , P and n are first chosen. Then, an entity selects an integer k uniformly at random from [1 , n – 1] and computes $Q = [k]P = \underbrace{P \oplus P \oplus \cdots \oplus P}_{k \text{ times}}$. The entity's public key is Q, while the entity's private key is k. The integer k is called the discrete logarithm of Q to the base P, denoted $k = \log_P Q$. Clearly, if the ECDLP is easy, then an adversary can deduce k from Q. Thus, the hardness of the ECDLP is crucial for the security of all ECC_s [6].

3 Proposing Method for Solving ECDLP

If $\mathbf{E}(F_p)$ is cyclic group of order n generated by P (i.e. $\mathbf{E}(F_p) = \langle P \rangle$) then by using theorem in abstract algebra that statements ((Let G be a cyclic group with n elements and generated by a. let b ∈ G and $b = a^k$. then b generates a cyclic subgroup H of G containing $\frac{n}{d}$ elements, where d is greatest common divisor of n and k)) [2].
$\mathbf{E}(F_p)$ be a cyclic group with n elements and generated by P. let $Q \in \mathbf{E}(F_p)$ then Q generates a cyclic subgroup H, then by theorem above then ord (H) equal $\frac{n}{d}$ where d = g.c.d(n, k) and $1 \leq k < n - 1$. Suppose that k_i is the values which satisfying the form g.c.d(n, k_i) = d. We can now compute $[k_i]P$, for $1 \leq k_i \leq n - 1$ until we have found a match with point Q.

If $\mathbf{E}(F_p)$ is not cyclic group of order n generated by P (i.e. $\mathbf{E}(F_p) \neq \langle P \rangle$) then we consider $\langle P \rangle$ is cyclic group then by using theorem above. Let $Q \in \langle P \rangle$ then Q generates a cyclic subgroup H, then by theorem above then ord (H) equal $\frac{n}{d}$ where d = g.c.d(n, k) and $1 \leq k \leq n - 1$. Suppose that k_i is the values which satisfying the form g.c.d(n, k_i) = d. We can now compute $[k_i]P$, for $1 \leq k_i \leq n - 1$ until we have found a match with point Q. In this method we reduce the values of k from n – 1 to k_i values that satisfying the form g.c.d(n, k_i) = d, where $d = \frac{n}{\text{ord } Q} = \frac{\text{ord } P}{\text{ord } Q}$.

Algorithm 3.1

A proposed method algorithm for computing ECDLP.

INPUT: a generator P of a cyclic group $\langle P \rangle$, of order n and an point $Q \in \langle P \rangle$.
OUTPUT: the discrete logarithm $k = \log_P Q$.

1. Compute order of Q and $d = \frac{n}{\text{ord } Q}$
2. For i from 1 to ord Q
   2.1 set x =d*i
   2.2 if gcd(ord P , x) = d then k = x
   2.3 check if scalar multiplication to $[k_i]P = Q$
   2.4 ki = k
3. Return k

Example 3.1

Consider the elliptic curve **E** : $y^2 = x^3 + 9x + 2$ defined over F_{41}.

Let P = (7, 11) ∈ **E** (F_{41}).We wish to determine the discrete logarithm of point Q = (24, 15) to the base P.

Solution //

The order of P is 39. We compute order of Q is 3 since [3]Q = $\mathbf{O}_\infty$ then $d = \frac{\text{ord } P}{\text{ord } Q} = \frac{39}{3} = 13$.

g.c.d(n, k_i) = d ⇒ g.c.d.(39, k_i) = 13 where $1 \le k_i \le 38$ then k_i = 1 , 13 , 26.
Now compute [k_i]P until we have found a match with point Q.

k_i	[k_i]P	k_i	[k_i]P
1	(7, 11)	13	(24, 26)
26	(24, 15)		

Table 1 : Data for Proposing Method

Since [26]P = Q, then $k = \log_P Q = 26$.

Example 3.2

Consider the elliptic curve **E** : $y^2 = x^3 + 22x + 4$ defined over F_{307}.

Let P = (275, 84) ∈ **E** (F_{307}).We wish to determine the discrete logarithm of point Q = (247, 31) to the base P.

Solution

The order of P is 152. We compute order of Q is 19 since [19]Q = $\mathbf{O}_\infty$ then $d = \frac{\text{ord } P}{\text{ord } Q} = \frac{152}{19} = 8$.

g.c.d(n, k_i) = d ⇒ g.c.d.(152, k_i) = 8 where $1 \le k_i \le 151$ then k_i = 8 , 16 , 24 , 32 , 40 , 48 , 56 , 64 , 72 , 80 , 88 , 96 , 104 , 112 , 120 , 128 , 136 , 144.

Now compute [k_i]P until we have found a match with point Q.

k_i	[k_i]P	k_i	[k_i]P
8	(46, 172)	16	(281, 176)
24	(269, 28)	32	(20, 48)
40	(182, 124)	48	(77, 164)
56	(247, 276)	64	(273, 294)
72	(220, 78)	80	(220, 229)
88	(273, 13)	96	(247, 31)

Table 2 : Data for Proposing Method

Since [96]P = Q, then $k = \log_P Q = 96$.

Example 3.3

Consider the elliptic curve **E** : $y^2 = x^3 + 1x + 2$ defined over F_{1013}.

Let P = (1000, 638) ∈ **E** (F_{1013}).We wish to determine the discrete logarithm of point Q = (919, 833) to the base P.

Solution

The order of P is 340. We compute order of Q is 17 since [17]Q = $\mathbf{O}_\infty$ then $d = \frac{\text{ord } P}{\text{ord } Q} = \frac{340}{17} = 20$.

g.c.d(n, k_i) = d ⇒ g.c.d.(340, k_i) = 20 where $1 \le k_i \le 339$ then
k_i = 20 , 40 , 60 , 80 , 100 , 120 , 140 , 160 , 180 , 200 , 220 , 240 , 260 , 280.

Now compute [k_i]P until we have found a match with point Q.

k_i	$[k_i]P$	k_i	$[k_i]P$
20	(52, 442)	40	(205, 511)
60	(919, 180)	80	(272, 537)
100	(201, 852)	120	(570, 315)
140	(437, 45)	160	(331, 689)
180	(331, 324)	200	(467, 968)
220	(570, 698)	240	(201, 161)
260	(272, 476)	280	(919, 833)

Table 3 : Data for Proposing Method

Since $[280]P = Q$, then $k = \log_P Q = 280$.

4 Conclusions

This paper provides a new proposed method for solving the ECDLP. This method can be considered as a new approach to tackle the problem of attacking the ECDLP based on theorem in abstract algebra. In this proposed method we reduce the values of k from $n - 1$ to k_i values that satisfying the form $g.c.d(n, k_i) = d$, where $d = \frac{\text{ord } P}{\text{ord } Q}$. That is provides a reduction to mathematical operations. This leads to main conclusion that the new proposed method is batter than the Exhaustive Search in the reduction cost can be offered for complexity of calculation. As another result we can have an important criterion to choose a secret key to elliptic curve cryptosystems. When a users selects a generator $P \in \mathbf{E}(F_p)$, if the users want to select a good secret key k, then users must be select secret key k such that satisfy the relation $g.c.d(k , n) = 1$ or the users must be select P which have order prime. So, if n is prime then this method will be like Exhaustive Search which can take up n step in the worst case.

References

[1] A. Escott, J. Sager, A. Selkirk and D. Tsapakidis, "Attacking Elliptic Curve Cryptosystems Using The Parallel Pollard Rho Method", CryptoBytes–The Technical Newsletter of RSA Laboratories, volume 4, number 2, Winter 1999, 15-19.

[2] W. Fraleigh, "A First Course in Abstract Algebra", Third edition, Addison Wesley Publisher, 1982.

[3] S. D. Galbraith And N. P. Smart " Evaluation Report For Cryptrec: Security Level Of Cryptography–ECDLP Mathematical Problem ", 2001. http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1029_report.pdf.

[4] Koblitz, "A Course in Number Theory and Cryptography", Springer-Verlag, 2nd edition, 1994.

[5] A. Menezes, "Elliptic Curve Public Key Cryptosystems", Kluwer Academic Publishers, 1993.

[6] A. Menezes, "Evaluation of Security Level of Cryptography: The Elliptic Curve Discrete Logarithm Problem (ECDLP)", 2001. Technical Report http//www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1028 ecdlp.pdf.

[7] M. Wiener and R. Zuccherato, "Faster Attacks on Elliptic Curve Cryptosystems", Selected Areas in Cryptography, Lecture Notes in Computer Science, 1556, Springer-Verlag, 190-200, 1999.

Appendix 1

The Program for computing the discrete logarithm k of point Q = (x2, y2) to the base P = (x1, y1) from Q = [k]P, where P, $Q \in \mathbf{E} : y^2 = x^3 + ax + b$ defined over F_p.

```
(1)  % program to find secret key k
(2)  p  = input ('enter prime no. p =');
(3)  a  = input('enter integer no.a=');
(4)  b  = input('enter integer no.b=');
(5)  x1  = input ('enter integer no. x1=');
(6)  y1  = input ('enter integer no. y1=');
(7)  x2  = input ('enter integer no. x2 =');
(8)  y2  = input ('enter integer no. y2 =');
(9)  for k = 1:2*p
(10) r = dec2bin(k);[row,col] = size(r);
(11) xk = x1;
(12) yk = y1;
(13) for i = 2:col
(14) m1 = mod(3*xk^2+a,p);
(15) m2 = mod(2*yk,p);
(16) for z = 1:p-1
(17) w = mod(m2*z,p);
(18) if w = = 1;[z]; m = mod(m1*z,p);
(19) end, end
(20) x3 = mod(m^2-2*xk,p);
(21) y3 = mod(m*(xk-x3)-yk,p);
(22) s = [x3 y3];
(23) xk = s(1);
(24) yk = s(2);
(25) if r(i) = = 49
(26) m1 = mod(yk-y1,p);
(27) m2 = mod(xk-x1,p);
(28) for z = 1:p-1
(29) w = mod(m2*z,p);
(30) if w = = 1;[z]; n = mod(m1*z,p);
(31) end, end
(32) x4 = mod(n^2-x1-xk,p);
(33) y4 = mod(n*(x1-x4)-y1,p);
(34) z = [x4 y4] ;
(35) xk = z(1);
(36) yk = z(2);
(37) end, end
(38) if xk = = x1 & yk ~ = y1
(39) ordP = [k+1], break
(40) end
(41) R = [xk,yk];
(42) end
(43) for k = 1:2*p
(44) r = dec2bin(k);
(45) [row,col] = size(r);
(46) xk = x2;
(47) yk = y2;
(48) for i = 2:col
(49) m1 = mod(3*xk^2+a,p);
(50) m2 = mod(2*yk,p);
(51) for z = 1:p-1
(52) w = mod(m2*z,p);
(53) if w = = 1; [z] ; m = mod(m1*z,p);
(54) end, end
(55) x3 = mod(m^2-2*xk,p);
(56) y3 = mod(m*(xk-x3)-yk,p);
(57) s = [x3 y3];
(58) xk = s(1);
(59) yk = s(2);
(60) if r(i) = = 49
(61) m1 = mod(yk-y2,p);
(62) m2 = mod(xk-x2,p);
(63) for z = 1:p-1
(64) w = mod(m2*z,p);
(65) if w = = 1 ; [z] ; n = mod(m1*z,p);
(66) end, end
(67) x4 = mod(n^2-x2-xk,p);
(68) y4 = mod(n*(x2-x4)-y2,p);
(69) z = [x4 y4] ;
(70) xk = z(1);
(71) yk = z(2);
(72) end, end
(73) if xk = = x2 & yk ~ = y2
(74) ordQ = [k+1], break
(75) end
(76) R=[xk,yk];
(77) end
(78) d = ordP / ordQ
(79) ki = 0;
(80) for i = 1:ordQ
(81) x = d*i;
(82) if gcd(ordP,x) = = d
(83) k = x;
(84) r = dec2bin(k);
(85) [row,col] = size(r);
(86) xk = x1;
(103) for z = 1:p-1
```

```
(87)  yk = y1;
(88)  for i = 2:col
(89)  m1 = mod(3*xk^2+a,p);
(90)  m2 = mod(2*yk,p);
(91)  for z = 1:p-1
(92)  w = mod(m2*z,p);
(93)  if w = = 1;[z]; m = mod(m1*z,p);
(94)  end, end
(95)  x3 = mod(m^2-2*xk,p);
(96)  y3 = mod(m*(xk-x3)-yk,p);
(97)  s = [x3 y3];
(98)  xk = s(1);
(99)  yk = s(2);
(100) if r(i) = = 49
(101) m1 = mod(yk-y1,p);
(102) m2 = mod(xk-x1,p);
(104) w = mod(m2*z,p);
(105) if w = = 1;[z];n=mod(m1*z,p);
(106) end, end
(107) x4 = mod(n^2-x1-xk,p);
(108) y4 = mod(n*(x1-x4)-y1,p);
(109) z = [x4 y4];
(110) xk = z(1);
(111) yk = z(2);
(112) end, end
(113) [xk,yk];
(114) if [xk,yk] = = [x2,y2]
(115) ki = k; break
(116) end, end, end
(117) secretkey = k
```