New Random Walk Technique and Collision Detection Algorithm to Improve the Pollard RHO Attack of Solving Discrete Logarithm Problem on Elliptic Curves

CH. Suneetha[1],P. Sirisha[2], D. Sravana Kumar[3], KM sandeep[4]

[1] *Associate Professor in Mathematics, GITAM University,Visakhapatnam India*
[2] *Faculty in Mathematics, Indian Maritime University, Visakhapatnam, india*
[3] *Associate Professor in Physics, Dr VS Krishna Govt.Degree College, Visakhapatnam, India*
[4]*Department of Computer science and Engineering, GITAM university, Visakhpatnam, India*

Abstract- *Elliptic curve cryptography is a revolutionary in the history of public key cryptography that is protected by a hard problem Elliptic Curve Discrete Logarithm Problem (ECDLP). A wide research has been done on cryptanalysis of ECDLP. In 1978 Pollard developed an algorithm with a "Monte-Carlo" method for solving ECDLP called Pollard Rho attack which is the quickest algorithm. Since then the algorithm was modified to increase the efficiency of Pollard Rho algorithm in relatively short time to find the insecurity of the elliptic curve cryptosystem. The present paper designs a new random walk technique and collision detection algorithm to improve the performance of Pollard Rho algorithm.*

Key words: *Pollard Rho Algorithm, Random walk, Collision Detection*

I. INTRODUCTION

Invention of public-key cryptography is the mile stone in the history of cryptography. The principal focus of inventors of ECC [1,2] was the study of advantages of elliptic curve cryptography in wireless communications in place of well known traditional RSA cryptosystem. An affine equation E: $y^2 + b_1xy + b_3y = x^3 + b_2x^2 + b_4x + b_6$ over the set of real numbers is said to be weierstrass equation, where b_1, b_2, b_3 , b_4, b_6 and x, y are real numbers. An elliptic curve for cryptographic purpose is defined by the equation $y^2 = x^3 + ax + b$, $4a^3+27b^2 \# 0$ over the prime field F_p.

II. GROUP LAWS OF ELLIPTIC CURVE

Let E be an elliptic curve defined over the finite field of integers K. Addition of two points uses chord-and-tangent rule to get the third point [1,2,3]. The set of all points on the elliptic curve over the finite field with addition as binary operation forms an abelian group with ∞, the point at infinity as identity element.

A. Geometric Rules Of Addition

Let $P(x_1,y_1)$ and $Q(x_2,y_2)$ be two points on the elliptic curve E. The sum of the two points P and Q is $R(x_3,y_3)$ which is the reflection of the point of intersection of the line through the points P, Q and the elliptic curve about x axis. The same geometric interpretation also applies to two points P and –P, with the same x-coordinate. Here the points are joined by a vertical line, which is regarded as the intersecting point on the curve at the point infinity. $P + (-P) = \infty$, the identity element which is the point at infinity [1,2,3].

B. Doubling The Point On The Elliptic Curve

If $P(x_1,y_1)$is point on the elliptic curve then 2P is the reflection of the point of intersection of the tangent line at P and the elliptic curve about x axis [1,2,3]. Example of addition of two points and doubling of a point are shown in the following figures 1 and 2 for the elliptic curve $\mathbf{y^2 = x^3-x}$.

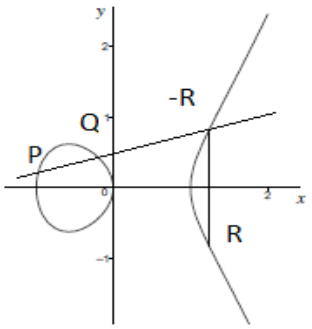

Figure 1. Geometric addition

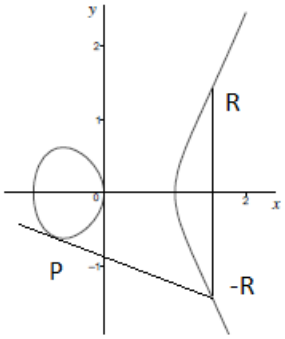

Figure 2. Geometric doubling

Identity:-$P + \infty = \infty + P = P$ for all E where ∞ is the point at infinity.

Negatives: - If P(x,y) is a point on the elliptic curve then $(x,y) + (x,-y) = \infty$. Where $(x, -y)$ is the negative of P denoted by –P.

Point addition:- If $P(x_1,y_1)$, $Q(x_2,y_2)$ are two points where $P \neq Q$. Then $P + Q = (x_3,y_3)$ [1,2]where

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \text{ and}$$

$$y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1$$

Point Doubling:- Let $P(x_1,y_1) \in E(K)$ where $P \neq -P$ then

$$2P = (x_3, y_3) = [1,2] \text{ where } x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \text{ and } y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1$$

Point Multiplication:-Let P be any point on the elliptic curve over the finite field of integers. Then the operation multiplication of P is defined as repeated addition [17]. kP = P+P+……..k times

III Elliptic Curve Discrete logarithmic Problem

The strength of Elliptic Curve Cryptography depends on the hard problem known Elliptic Curve Discrete Logarithmic Problem [3,4,6,19,20]. Consider an equation Q = xP where $Q,P \in E(F_q)$ the elliptic curve over the finite field and $x \in [1,2,\ldots\ldots q-1]$. It is relatively easy to calculate Q for given x and P. But it is relatively hard to determine $x = \log_P Q$ given Q and P .

A wide research has been done on cryptanalysis of ECDLP [3]. In 1978 Pollard developed an algorithm using "Monte-Carlo" method to solve ECDLP called Pollard Rho attack which is a random method to compute discrete logarithm. Pollard Rho algorithm can be applied to cyclic groups. In group theory baby step giant step is a meet – in – middle algorithm to compute discrete logarithm. It is an ignorant method of finding discrete logarithm depending on space time tradeoff. Unlike baby step giant step Pollard Rho algorithm is more effective to complete the mission of finding DLP within the expected time of execution and with high probability. The design of the algorithm is to produce [5] a sequence of randomly generated terms (X_i,a_i,b_i) where X_i is the point on the elliptic curve $E(F_q)$. As $E(F_q)$ is a finite cyclic group the sequence finally converts to a periodic sequence and comes back to earlier term in the sequence. This periodic property helps to break the hardness of ECDLP in this algorithm. The cycle always does not start from the first term. So, the diagram of the sequence is similar to a Greek letter ρ and is called Pollard Rho algorithm. Here the number of iterations before the cycle starts is called tail length (t), the number of iterations after the cycle starts until the collision occurs is called cycle length (s).

A. *Pollard Rho Algorithm [9,13]*

Consider an elliptic curve over the prime field $E(F_q)$ where the order $\#E(F_q) = n$ and two points R,S such that $S = xR$. ECDLP is to find out $x = \log_R S$. The following steps are involved in the algorithm:

1. Set of points in the group are classified into three disjoint sets A_1, A_2, A_3 roughly of the same size using a hash function .

2. An iteration function f of random walk is defined as $f(B_i) = B_{i+1} = \left\{ \begin{array}{l} S + B_i, B_i \in A_1 \\ 2B_i, B_i \in A_2 \\ R + B_i, B_i \in A_3 \end{array} \right\} \quad B_i = r_i R + s_i S$

Where r_i, s_i are two random integers in [0,1,2,…….n-1]. The random sequence of integers r_i, s_i are determined by the iteration function

$$r_{i+1} = \left\{ \begin{array}{l} r_i, B_i \in A_1 \\ 2r_i \pmod n, B_i \in A_2 \\ r_i + 1, B_i \in A_3 \end{array} \right\}$$

$$s_{i+1} = \left\{ \begin{array}{l} s_i + 1, B_i \in A_1 \\ 2s_i \pmod n, B_i \in A_2 \\ s_i, B_i \in A_3 \end{array} \right\}$$

3. Using iteration function two points Bj, B2j are computed at random until the collision occurs.

4. If two points Bj = B2j is obtained for $j \neq 2j$ then x value is calculated as

$$x = \frac{r_{2j} - r_j}{s_j - s_{2j}} \pmod n$$

Using this algorithm the value of x can be determined provided $\gcd[(s_j - s_{2j}),n]=1$

Here the number of iterations [13] for the occurrence of the collision is approximately equal to $\sqrt{\pi n/2}$. If n is a prime the probability of the positive result of the attack increases. In Pollard Rho algorithm the iteration function of random walk for finding random sequence of points is determined by probability distribution function. The following modifications have been done to boost up the parameters of Pollard Rho algorithm:

1. *Increasing the number of partitions*: In original algorithm the group elements are divided into three disjoint sets using a hash function. The partition technique was not clearly mentioned in the algorithm [4]. So, no straight forward method is there for splitting the group elements.

2. *Existing methods for partitioning of the group elements and Iteration function of random walks*:

i. Depending on the order of the group # E [F_q] the elements are divided into three disjoint classes A_1,A_2,A_3 roughly of the same size with y co-ordinate value between $[0,\frac{q}{3})$ $[\frac{q}{3},\frac{2q}{3})$, $[\frac{2q}{3},q)$ successively [5,6].

ii. By reducing x co-ordinate or y co-ordinate to modulo 3 and making arbitrary choice to place ∞ the elements are divided into three disjoint classes [5,6].

iii. Teske [7, 8,13,14,18] suggested that two factors play vital role in the overall performance of the Pollard Rho algorithm. First one increase of the number of partitions which increases the randomness of the iteration function. Second one enhancing the addition walks and point doubling operations. He recommended to classify the group elements into g disjoint sets $A_1,A_2\ldots\ldots\ldots A_g$ roughly of the same size by reducing the x coordinate value mod g. Moreover he proposed and implemented an r adding walk and g+h mixed walk where g is the number of point additions, h the number of point doubling operations. For this the elements are divided into g small subsets for g adding walk, g+h small subsets for g+h random walk. Also he defined a hash function v : G $\rightarrow$ {1,2,3,…..g}, used a rational approximation of

$$(r_i, s_i) = \{(r + r_j) + (s + s_j)\} if B_i \in A_j .$$

Storehouse [8,9,10,11,15,16]for the generated elements by iteration function is a major concern in Pollard Rho algorithm . Research is conducted on this issue to find the matching of two points as quickly as possible using Floyd's cycle detection algorithm called hair and tortoise algorithm [10], Brent's cycle detection algorithm using two pointer techniques and Nivasch's cycle finding algorithm using stack. In Nivasch's cycle finding algorithm [10,11,12]a stack is generated and begins up empty. In each step j all the uppermost entries (r_i, i) are removed (pop) from the stack for $r_i > r_j$. This process is continued until two points match each other. i.e., $r_i = r_j$ for $i \neq j$ so that the cycle length is j-i. Otherwise the entry (r_j, j) is appended to the stack (push). Basing on this cycle finding algorithm Pollard Rho algorithm is modified. The primary idea here is to use a stack as storehouse for the elements generated by iteration function. Here the pairs (i, r_i, s_i, B_i) are reserved in the stack with increasing sequence of B_i in lexicographic order on E(F_q) . If $B_j > B_i$ for $0 < i < j$ then the pair (j, r_j, s_j, B_j) is added (push) into the stack. Otherwise the top most entries (i, r_i, s_i, B_i) are removed (pop) from the stack until the collision or match occurs. i.e., $B_i = B_j$ for $i \neq j$. This stack algorithm requires n(n+1)|2 iterations [10,12] to determine the matching of two points where n is the number of elements in the group.

IV. Proposed method

The principal architecture of the present paper is to improve the performance of Pollard Rho algorithm in view of the major factors partitioning of the group elements, usage of more effective iteration function than what the earlier researchers used and easier as well as faster collision detection algorithm so that the hard problem ECDLP is weakened.

A. Partition of the group elements

Increasing the number of partitions increases the calibre of
randomness of the iteration function. Basing on the literature division techniques of group elements in original Pollard Rho algorithm here also the elements on the elliptic curve E(F_q) are classified into three disjoint classes A_1,A_2,A_3 roughly of the same size. Since the elements on the elliptic curve are symmetric about x- axis the

disjoint classes A_1,A_2,A_3 are further divided into three subclasses each. i.e, A_1 is divided into subclasses A_{11}, A_{12}, A_{13}; A_2 is divided into A_{21}, A_{22}, A_{23} ; A_3 is divided into A_{31}, A_{32}, A_{33}

Iteration function of Random walk :

1. The group elements are divided into disjoint classes A_{11}, A_{12}, A_{13} ; A_{21}, A_{22}, A_{23}; A_{31}, A_{32}, A_{33}
2. Let $B_i = r_i R + s_i S$ where R,S are points on the elliptic curve $E(F_q)$ and r_i, s_i are two random numbers in [0,1,2,…….n-1].An iteration function f of random walk is defined as

$$f(B_i) = B_{i+1} = \begin{Bmatrix} \begin{Bmatrix} B_i + S, B_i \in A_{11} \\ B_i + 2S, B_i \in A_{12} \\ B_i + 3S, B_i \in A_{13} \end{Bmatrix} \\ \begin{Bmatrix} 2B_i, B_i \in A_{21} \\ 3B_i, B_i \in A_{22} \\ 4B_i, B_i \in A_{23} \end{Bmatrix} \\ \begin{Bmatrix} B_i + R, B_i \in A_{21} \\ B_i + 2R, B_i \in A_{32} \\ B_i + 3R, B_i \in A_{33} \end{Bmatrix} \end{Bmatrix}$$

The sequence of integers r_i, s_i are determined by the iteration function

$$r_{i+1} = \begin{Bmatrix} \begin{Bmatrix} r_i, B_i \in A_{11} \\ 2r_i, B_i \in A_{12} \\ 3r_i, B_i \in A_{13} \end{Bmatrix} \\ \begin{Bmatrix} 2r_i (\mathrm{mod}\, n), B_i \in A_{21} \\ 4r_i (\mathrm{mod}\, n), B_i \in A_{22} \\ 6r_i (\mathrm{mod}\, n), B_i \in A_{23} \end{Bmatrix} \\ \begin{Bmatrix} r_i + 1, B_i \in A_{31} \\ r_i + 2, B_i \in A_{32} \\ r_i + 3, B_i \in A_{33} \end{Bmatrix} \end{Bmatrix} \qquad s_{i+1} = \begin{Bmatrix} \begin{Bmatrix} s_i + 1, B_i \in A_{11} \\ s_i + 2, B_i \in A_{12} \\ s_i + 3, B_i \in A_{13} \end{Bmatrix} \\ \begin{Bmatrix} 2s_i (\mathrm{mod}\, n), B_i \in A_{21} \\ 4s_i (\mathrm{mod}\, n), B_i \in A_{22} \\ 6s_i (\mathrm{mod}\, n), B_i \in A_{23} \end{Bmatrix} \\ \begin{Bmatrix} s_i, B_i \in A_{31} \\ 2s_i, B_i \in A_{32} \\ 3s_i, B_i \in A_{33} \end{Bmatrix} \end{Bmatrix}$$

3. Two points B_j,B_{2j} are computed at random until the collision of two points occurs If $B_j = B_{2j}$ for $j \neq 2j$ x value is calculated for $b_j \neq b_{2j}$

$$x = \frac{r_{2j} - r_j}{s_j - s_{2j}} (\mathrm{mod}\, n)$$

Algorithm for finding iteration function

1: function $f(B_i):B_{i+1}$ where B_i is initial point
2: if $B_i \in A_{11}$ then
3: $B_{i+1} \leftarrow S + B_i$
4: else if $B_i \in A_{12}$ then
5: $B_{i+1} \leftarrow 2S + B_i$

6: else if $B_i \in A_{13}$ then
7: $B_{i+1} \leftarrow 3S + B_i$
8: else if $B_i \in A_{21}$ then
9: $B_{i+1} \leftarrow 2\ B_i$
10: else if $B_i \in A_{22}$ then
11: $B_{i+1} \leftarrow 3\ B_i$
12: else if $B_i \in A_{23}$ then
13: $B_{i+1} \leftarrow 4\ B_i$
14: else if $B_i \in A_{31}$ then
15: $B_{i+1} \leftarrow R + B_i$
16: else if $B_i \in A_{32}$ then
17: $B_{i+1} \leftarrow 2R + B_i$
18: else if $B_i \in A_{33}$ then
19: $B_{i+1} \leftarrow 3R + B_i$
20: end if
21: return B_i
22: end function

Algorithm for finding (r_{i+1}, s_{i+1})

1: function $r_{i+1} = f(r_i)$; $s_{i+1} = f(s_i)$
2: if $B_i \in A_{11}$ then
3: $r_{i+1} \leftarrow r_i$, $s_{i+1} \leftarrow s_i + 1$
4: else if $B_i \in A_{12}$ then
5: $r_{i+1} \leftarrow 2r_i$, $s_{i+1} \leftarrow s_i + 2$
6: else if $B_i \in A_{13}$ then
7: $r_{i+1} \leftarrow 3r_i$, $s_{i+1} \leftarrow s_i + 3$
8: else if $B_i \in A_{21}$ then
9: $r_{i+1} \leftarrow 2r_i$, $s_{i+1} \leftarrow 2s_i$
10: else if $B_i \in A_{22}$ then
11: $r_{i+1} \leftarrow 4r_i$, $s_{i+1} \leftarrow 4s_i$
12: else if $B_i \in A_{23}$ then
$r_{i+1} \leftarrow 6r_i$, $s_{i+1} \leftarrow 6s_i$
13: else if $B_i \in A_{31}$ then
14: $r_{i+1} \leftarrow r_i + 1$, $s_{i+1} \leftarrow s_i$
15: else if $B_i \in A_{32}$ then
16: $r_{i+1} \leftarrow r_i + 2$, $s_{i+1} \leftarrow 2s_i$
17: else if $B_i \in A_{33}$ then
18: $r_{i+1} \leftarrow r_i + 3$, $s_{i+1} \leftarrow 3s_i$
19: end if
20: return r_i, s_i..

Algorithm for Collision Detection algorithm

Require: R,S,(A_{11},A_{12},A_{13}), (A_{21},A_{22},A_{23}), (A_{31},A_{32},A_{33})
Ensure: Integer 1 where S =xR

1: $B_i \leftarrow r_i R + s_i S$ where i is integer
1: $a_i \leftarrow$random $\in]0; n[$, i is integer
2: $b_i \leftarrow$random $\in]0; n[$, i is integer
3: $m \leftarrow 0$
4: $B_0 \leftarrow r_0 R + s_0 S$
5: for all m if $B_m \neq B_{2m}$ do
6: $(B_{m+1}, r_{m+1}, s_{m+1}) \leftarrow f(B_m), f(r_m, s_m)$

7: $(B_{2(m+1)}, r_{2(m+1)}, s_{2(m+1)}) \leftarrow f(f(B_{2m})), f(f(r_{2m}, s_{2m}))$
8: m: m+1
9: if $B_m = B_{2m}$ and $s_m \neq s_{2m}$ then
10: $X \leftarrow \frac{r_{2m}-r_m}{s_m-s_{2m}} \bmod (n)$
11: else if $s_m = s_{2m}$ then
12: $r_0 \leftarrow$ random $\in]0; n[$
13: $s_0 \leftarrow$ random $\in]0; n[$
14: $m \leftarrow 0$
15: end if
16: end for
17: return x

Example 1 comparison with original Pollard Rho algorithm:

Consider an elliptic curve E (F_{37}) $y^2=x^3+2x+9$ over the prime field F_{37}

There are 43 points on the curve
{0,(5,25),(1,30)(21,32),(7,25),(25,12),(4,28),(0,34),(16,17),(15,26),(27,32),(9,4),
(2,24),(26,5),(33,14),(11,17),(31,22),(13,30),(35,21),(23,7),(10,17),(29,6),(29,31),(10,20),
(23,30),(35,16),(13,7),(31,15), (11,20),(33,23),(26,32), (2,13),(9,33),(27,5),(15,11),(16,20),(0,3),(4,9),(25,25),
(7,12),(21,5), (1,7), (5,12)}

Since p = 37, basing on the value of y co-ordinate the points are classified into 3 disjoint sets A_1, A_2, A_3 as follows:

A_1 = {0,(5,25),(1,30)(21,32),(7,25),(25,12),(4,28), (0,34),(16,17),(15,26),(27,32),(9,4),(2,24),(26,5)}

A_2 = (33,14),(11,17),(31,22),(13,30),(35,21),(23,7),(10,17), (29,6),(29,31),(10,20),(23,30),(35,16),(13,7),
(31,15)}

A_3={(11,20), (33,23), (26,32),(2,13),(9,33), (27,5),(15,11), (16,20),(0,3),(4,9),(25,25),(7,12),(21,5),(1,7),(5,12)}

Let P=(25,12),Q=(35,21),a_0=1,b_0=0, B_0 = P

Random sequence of points are computed using the iteration function of **the original Pollard Rho algorithm** $f(B_i) = B_{i+1} = \begin{cases} Q + B_i, B_i \in A_1 \\ 2B_i, B_i \in A_2 \\ P + B_i, B_i \in A_3 \end{cases}$

$$a_{i+1} = \begin{cases} a_i, B_i \in A_1 \\ 2a_i \pmod n, B_i \in A_2 \\ a_i + 1, B_i \in A_3 \end{cases} \quad b_{i+1} = \begin{cases} b_i + 1, B_i \in A_1 \\ 2b_i \pmod n, B_i \in A_2 \\ b_i, B_i \in A_3 \end{cases}$$

i	B_i	a_i	b_i
0	(25,12)	1	0
1	(10,20)	1	1
2	(21,32)	2	2
3	(29,6)	2	3
4	(5,12)	4	6
5	(7,25)	5	6
6	(29,31)	5	7
7	(5,25)	10	14

8	(23,7)	10	15
9	(25,25)	20	30
10	(0,0)	21	30
11	(35,21)	21	31
12	(0,3)	5	25
13	(1,7)	6	25
14	(21,32)	7	25

Since $B_2 = B_{14}$ =(21,32) then $[a_2]P + [b_2]Q = [a_{14}]P + [b_{14}]Q$

$\Longrightarrow$ 2(25,12) +2(35,21) = 7(25,12) + 25(35,21) which gives

$X = \frac{a_{2m}-a_m}{b_m-b_{2m}} = \frac{7-2}{2-25} = (5).(14)^{-1} \bmod 37 = 3$ Therefore $X = \text{Log}_P Q = 30$

Now using the proposed algorithm the group elements are divided into disjoint classes
A_{11}={0,(5,25),(1,30)(21,32),(7,25)} A_{12}={(25,12),(4,28),(0,34),(16,17),(15,26)}
A_{13}={(27,32),(9,4),(2,24),(26,5)} A_{21}={(33,14),(11,17),(31,22),(13,30),(35,21)}
A_{22} = {(23,7),(10,17),(29,6),(29,31),(10,20)}
A_{23} = {(23,30), (35,16),(13,7),(31,15)}
A_{31} = {(11,20), (33,23), (26,32),(2,13),(9,33)}
A_{32} = {(27,5),(15,11),(16,20),(0,3),(4,9)}
A_{33} = {(25,25),(7,12),(21,5),(1,7),(5,12)}.
The random walk is performed using the newly designed iteration algorithm

$$f(B_i) = B_{i+1} = \begin{Bmatrix} \begin{Bmatrix} B_i + Q, B_i \in A_{11} \\ B_i + 2Q, B_i \in A_{12} \\ B_i + 3Q, B_i \in A_{13} \end{Bmatrix} \\ \begin{Bmatrix} 2B_i, B_i \in A_{21} \\ 3B_i, B_i \in A_{22} \\ 4B_i, B_i \in A_{23} \end{Bmatrix} \\ \begin{Bmatrix} B_i + P, B_i \in A_{21} \\ B_i + 2P, B_i \in A_{32} \\ B_i + 3P, B_i \in A_{33} \end{Bmatrix} \end{Bmatrix}$$

The sequence of integers a_i,b_i are determined by the iteration function

$$a_{i+1} = \begin{Bmatrix} \begin{Bmatrix} a_i, B_i \in A_{11} \\ 2a_i, B_i \in A_{12} \\ 3a_i, B_i \in A_{13} \end{Bmatrix} \\ \begin{Bmatrix} 2a_i \pmod n, B_i \in A_{21} \\ 4a_i \pmod n, B_i \in A_{22} \\ 6a_i \pmod n, B_i \in A_{23} \end{Bmatrix} \\ \begin{Bmatrix} a_i + 1, B_i \in A_{31} \\ a_i + 2, B_i \in A_{32} \\ a_i + 3, B_i \in A_{33} \end{Bmatrix} \end{Bmatrix} \quad b_{i+1} = \begin{Bmatrix} \begin{Bmatrix} b_i + 1, B_i \in A_{11} \\ b_i + 2, B_i \in A_{12} \\ b_i + 3, B_i \in A_{13} \end{Bmatrix} \\ \begin{Bmatrix} 2b_i \pmod n, B_i \in A_{21} \\ 4b_i \pmod n, B_i \in A_{22} \\ 6b_i \pmod n, B_i \in A_{23} \end{Bmatrix} \\ \begin{Bmatrix} b_i, B_i \in A_{31} \\ 2b_i, B_i \in A_{32} \\ 3b_i, B_i \in A_{33} \end{Bmatrix} \end{Bmatrix}$$

Let P=(25,12),Q=(35,21),a_0=1,b_0=0, B_0 = P

I	B_i	a_i	b_i
0	(25,12)	1	0
1	(11,17)	2	2
2	(26,32)	4	4
3	(1,7)	5	4
4	(31,22)	8	12
5	(9,33)	16	24
6	(0,0)	17	24
7	(25,12)	17	25

Since $B_0 = B_7$ =(25,12) then $[a_0]\,P + [b_0]Q = [a_7]\,P + [b_7]Q$
$\Rightarrow$ 1(25,12) +0(35,21) = 17(25,12) + 25(35,21) which gives
$X = \frac{a_{2m}-a_m}{b_m-b_{2m}} = \frac{17-1}{0-25} = (16).(-24)^{-1} \bmod 37 = 10$

Example 2

Consider the elliptic curve $E(F_{97})$ $y^2=x^3-7x+10$ over F_{97}

The points on the curve are {0,(1,2),(1,95),(2,2),(2,95),(3,4),(3,93),(5,10),(5,87),(9,26), (9,71), (11,10),(11,87), (13,46), (13,51),(15,46),(15,51), (19,25),(19,72),(21,43),(21,54),(23,45),(23,52),(24,38), (24,59),(29,25),(29,72),(31,22), (31,75), (36,40), (37,35), (37,62),(40,1),(40,96),(41,29),(41,68),(43,8),(43,89), (44,37),(44,60),(46,11),(46,86),(49,25),(49,72),(51,44), (51,53),(52,34),(52,63),(53,3),(53,94),(54,21),(54,76),(55,18),(55,79),(63,36),(63,61),(64,0),(64,97),(67,34),(67, 63), (69,46),(69,51),(73,15),(73,82),(74,41),(74,56),(75,34),(75,63),(79,30)(79,67),(80,39),(80,58),(81,10),(81,87),(8 7,27), (87,70),(94,2),(94,95),(95,4),(95,93),(96,4), (96,93)}

Let a_0 =2,b_0=3,P=(23,45);Q=(43,89),R_0=(13,51)

Basing on x co-ordinate value the points on the curve are divided into disjoint classes
A_1 = {(3,4), (3,93), (9,26), (9,71), (15,46), (15,51), (21,43), (21,54),(24,38),(24,59),(36,40),(36,57),(51,44),(51,53), (54,21), (54,76), (63,36), (63,61), (69,46), (69,51), (75,34), (75,63), (81,10),(81,87),(87,27),(87,70),(96,4),(96,93)}

A_2 ={(1,2), (1,95), (13,46), (13,51), (19,25), (19,72), (31,22), (31,75), (37,35), (37,62), (40,1), (40,96), (43,8), (43,89), (46,11), (46,86), (49,25), (49,72), (52,34), (52,63), (55,18), (55,79),(64,0),(64,97),(67,34),(67,63), (73,15), (73,82), (79,30),(79,67),(94,2),(94,95)}

A_3 = {(2,2), (2,95), (5,10), (5,87), (11,10), (11,87), (23,45), (23,52), (29,25), (29,72), (41,29), (41,68), (44,37), (44,60), (53,3),(53,94),(74,41),(74,56),(80,39),(80,58),(95,4), (95,93)}

The random walk using original Pollard Rho algorithm is

I	B_i	a_i	b_i
0	(13,51)	2	3
1	(80,39)	4	6
2	(19,72)	5	6
3	(95,93)	10	12
4	(1,2)	11	12
5	(96,93)	22	24
6	(51,44)	22	25

7	(21,54)	22	26
8	(69,46)	22	27
9	(80,58)	22	28
10	(44,37)	23	28
11	(55,18)	24	28
12	(3,93)	48	56
13	(19,25)	48	57
14	(95,4)	96	114
15	(79,30)	97	114
16	(3,93)	194	228
17	(19,25)	194	229

Since $B_{13} = B_{17}$=(19,25) then $[a_{13}]\ P + [b_{13}]Q = [a_{17}]\ P + [b_{17}]Q$

$\Rightarrow$ 48(23,45) +57(43,89) = 0(23,45) + 35(43,89) which gives

$X = \frac{a_{2m}-a_m}{b_m-b_{2m}} = \frac{0-48}{57-35}$ = (-48).$(22)^{-1}$ mod 97 = (-48).(75)mod 97=-3600 mod 97= 86

Therefore $X = \text{Log}_p Q$ = 86

The random walk using the newly designed algorithm is

A_{11}={(9,26),(9,71),(36,40),(36,57),(54,21),(54,76),(63,36),(63,61),(81,10),(81,87)}

A_{12}={(1,2),(1,95),(19,25),(19,72),(37,35),(37,62),(46,11),(46,86),(55,18),(55,79),(64,0),(64,97),(73,15),(73,82)}

A_{13}={(2,2),(2,95),(11,10),(11,87),(29,25),(29,72),(74,41),(74,56)}

A_{21}={(3,4),(3,93),(21,43),(21,54),(75,34),(75,63)}

A_{22}={(13,46),(13,51),(31,22),(31,75),(40,1),(40,96), (49,25),(49,72),(67,34),(67,63),(94,2),(94,95)}

A_{23}={(5,10),(5,87),(23,45),(23,52),(41,29),(41,68),(95,4), (95,93)}

A_{31}={(15,46),(15,51),(24,38),(24,59),(51,44),(51,53),(69,46),(69,51),(87,27),(87,70),(96,4),(96,93)}

A_{32}={(43,8),(43,89),(52,34),(52,63),(79,30),(79,67)}

A_{33}={(44,37),(44,60),(53,3),(53,94),(80,39),(80,58)}

I	B_i	a_i	b_i
0	(13,51)	2	3
1	(74,41)	8	12
2	(21,54)	24	15
3	(80,39)	36	30
4	(69,51)	39	90
5	(36,83)	40	90
6	(54,21)	41	90
7	(53,94)	41	91
8	(74,41)	44	273

Since $B_1 = B_8$=(74,41) then $[a_1]\ P + [b_1]Q = [a_8]\ P + [b_8]Q$

$\Rightarrow$ 8(23,,45) +12(43,89) = 44(23,45) + 273(43,89) which gives

$X = \frac{a_{2m}-a_m}{b_m-b_{2m}} = \frac{44-8}{12-273}$= (36).$(261)^{-1}$ mod 97 = 40

Therefore $X = \text{Log}_P Q = 40$

S.No.	Example	No. of iterations in original Pollard algorithm	Collision detection in original Pollard Rho algorithm	No. of iterations in Proposed algorithm	Collision detection in proposed algorithm
1	1	14	$B_2 = B_{14}$	7	$B_0 = B_7$
2	2	17	$B_{13} = B_{17}$	8	$B_1 = B_8$

Proposed Collision Detection Algorithm: Stack is an ordinary method used for abstract data type in many programming languages associated with two operations push and pop. When compared to original Pollard Rho algorithm cycle detection algorithm using stack requires less number of iterations. Stack with n iterations uses 40n bytes. The number of iterations for n items in stack method is n(n+1)|2. To achieve the better performance for collision detection here we propose a new technique using insertion sorting rather than stack method. In data structures insertion sorting is a simple sorting algorithm works similar to the procedure of playing cards. This sorting technique is as efficient as advanced sorting algorithms like quicksort, heapsort and practically more efficient with time complexity log(n). Suppose some unsorted numbers are to be sorted in ascending order, the second element of the array is compared with the first element. If it is smaller than the first element then it is placed in the position of the first element. Next the third element is compared with the prior elements. If it is smaller than first one then it is inserted in the first element's position. If it is smaller than second and larger than first then it is placed in the second position. The process iterates further until the elements are arranged in the ascending order.

Example: Consider an unsorted array 9,5,2,4,7. It is sorted as

9	5	2	4	7
5	9	2	4	7
2	5	9	4	7
2	5	4	9	7
2	4	5	9	7
2	4	5	9	7
2	4	5	7	9

The same technique is applied here for fast collision detection. Insertion sorting is performed depending on Y co-ordinate. The Y co-ordinate values are sorted in the ascending order. The x co-ordinate value and the index of the point (B_0,B_1,B_2……) are also changed accordingly along with their Y co-ordinate value.
Example: Consider the points generated by iteration function $B_0 = (2,3)$ $B_1 = (2,1)$ $B_2 = (3,5)$
$B_3 = (5,4)$ $B_4 = (9,3)$ $B_5 = (3,5)$

y	3
x	2
i	0

B_0 =

y	1	3
x	2	2
i	1	0

B_1 =

y	1	3	5
x	2	2	3
i	1	0	2

B_2 =

y	1	3	4	5
x	2	2	5	3
i	1	0	3	2

B_3 =

B_4 =

y	1	3	4	5	9
x	2	2	5	3	3
i	1	0	3	2	4

y	1	3	4	5	5	9
x	2	2	5	3	3	3
i	1	0	3	2	5	4

B_5 =

Then $B_2 = B_5$

Algorithm for Insertion Sorting:

1: start
2:Initialize i = o, flag = 0
3:Loop starts if (flag = = 0)
4:calculate R_i and result stores in K where K is the point K = (x_i,y_i)
5: Insertion sort of (K,i) at position P
6:check p and (p-1) positions are equal or not [Data in p^{th} position, data in $(p-1)^{th}$ position]
7:If [D(p) = = D(p-1)] flag = 1
8:i = i++
9:loop ends
10: stop

Conclusions: in original Pollard Rho algorithm for random function f both tail and cycle length is expected $\sqrt{\frac{\pi n}{2}}$ where n is the order of the elliptic curve. So, two points match each other in $\sqrt{\frac{\pi n}{2}}$ iterations. So,30 to 35% more number of iterations than the expected number is required for the occurrence of the collision of two points. This is because the random walks in the iteration function are not sufficient enough to meet the collision. Research specifies that increasing the number of partitions, addition walks and point doubling operations enhances the fastness of random walk to meet the collision quickly, so that the quality of the algorithm will also be improved. Here in the proposed algorithm the subsets A_1,A_2,A_3 are further divided into three disjoint classes each. In addition the iteration function is modified by increasing the number of addition walks and point doubling operations. The speed of the iteration function is shown clearly in the table for examples 1 and 2. In example 1 the random walk using original Pollard Rho algorithm finds the matching after 14 iterations (tail length 3, cycle length 11) for the elliptic curve E (F_{37}) $y^2=x^3+2x+9$ where # E (F_{37}) = 43. i.e., $B_2 = B_{14}$. For the same curve using the proposed method the collision is found just for 7 iterations. i.e, $B_0 = R_7$ (tail length 0, cycle length 7)0. In the second example for the elliptic curve E(F_{97}) $y^2=x^3-7x+10$ where # E(F_{97}) = 82, original algorithm finds the matching after doing 17 iterations whereas the proposed algorithm detects the collision for 9 iterations $B_1 = B_8$ (tail length 1, cycle length 8). These two examples comprehensively explain the improvement of the Pollard Rho algorithm over the original algorithm. The present paper also explains a new technique of collision detection using insertion sorting. Literature on Pollard rho modifications state that the performance of the algorithm is improved by using stack for collision detection. But in stack method at each step only one (top most point) is compared for push or pop which requires n(n+1)|2 iterations in the best case. Besides insertion sorting for collision detection has linear running time $O(n)$ for the best case and quadratic running time $O(n^2)$ for the worst case. In original pollard Rho algorithm the pair (B_i,B_{2i}) is stored and in each step a new pair is generated, compared with the earlier pair and the earlier pair is discarded. But in this new collision detection technique using insertion sorting technique all the generated points B_i are stored with increasing sequence of y co-ordinate. No element is deleted from the array but every element is compared with previous element for matching. So, this paper presents a new technique for partitioning of the group elements with better iteration function of random walk. Moreover a new procedure of collision detection using insertion sorting technique is suggested. The two proposed techniques together will speed-up the performance of original Pollard Rho algorithm.

REFERENCES

[1] Koblitz N., "Elliptic curve cryptosystems, mathematics of computation", Vol. 48, No. 177, pp. 203-209, January 1987.
[2] Miller V., "Uses of elliptic curves in cryptography". In advances in Cryptography (CRYPTO 1985), Springer LNCS, 1985, vol. 218, pp 417-4 26.
[3] Maurer U., A. Menzes and E. Teske, "Analysis of GHS weil decent attack on the ECDLP over characteristic two fields of composite degree". LMS journal of computation and Mathematics, 5:127-174, 2002.
[4] Washington, Lawrence C., "Elliptic Curves: Number theory and cryptography", Chapman and Hall, Boca Raton, FL, 2nd. Ed., 2008.
[5] Lamb Nicholos, "An Investigation into Pollard Rho method for attacking Elliptic curve cryptosystems", 2002, http://www.cs.ualberta
[6] Arron Blumenfeld, "Discrete logarithms on Elliptic curves", 2011
[7] Ping Wang and Fangguo Zhang," An Efficient Collision Detection Method for Computing Discrete Logarithms with Pollard's Rho", Journal of Applied MathematicsVolume 2012 (2012), Article ID 635909, 15 pages.
[8] Ammar Ali Neamah,"New Collisions to Improve Pollard's Rho Method of Solving the Discrete Logarithm Problem on Elliptic Curves" Journal of Computer Science, Volume 11, Issue 9.

[9] Masaya Yasuda, Tetsuya Izu, Takeshi Shimoyama and Jun Kogure," On random walks of Pollard's rho method for the ECDLP on Koblitz curves", Journal of Math-for-Industry, Vol. 3 (2011B-3),
[10] Jung HeeCheon, Jin Hong, and Minkyu Kim," Speeding Up the Pollard Rho Method on Prime Fields".
[11] Paul C. van Oorschot and Michael J. Wiener." Parallel Collision Search with Application to Hash Functions and Discrete Logarithms".
[12] Mandy Zandra Seet, "ELLIPTIC CURVE CRYPTOGRAPHY Improving the Pollard-Rho Algorithm", https://www.maths.unsw.edu.au/sites/default/files/mandyseetthesis.pdf
[13] Teske, E., "On random walks for Pollard's RHO Method". Mathem. Comput., 70: 809-825, 2001.
[14] Anjan K Koundinya, Harish G, Srinath NK et.al, Parallelization of Pollard's Rho algorithm, proceeding of CCSEIT 2012 published in ACM-ICPS, pp. 43 -47, Oct 2012.
[15] Anjan K Koundinya, Harish G et.al, "performance analysis of parallel Pollard's Rho factoring algorithm", International Journal of Computer Science & Information Technology (IJCSIT) Vol 5, No 2, April 2013
[16] Kim, J. H., Montenegro, R. & Tetali, P. (2007), "A near optimal bound for Pollard's rho to solve discrete log", IEEE Proc. of the Foundations of Computer Science (FOCS), 2007, Providence, RI, to appear.
[17] D. J. Bernstein and T. Lange. "Analysis and optimization of elliptic-curve single scalar multiplication in Finite Fields and Applications", volume 461 of Contemporary Mathematics Series, pages 1-19, 2008.
[18] F. Kuhn and R. Struik "Random walks revisited: Extensions of Pollard's rho algorithm for computing multiple discrete logarithms". Selected areas in cryptography SAC 2001 (LNCS 2259) [468]212-229 2001.
[19] Menzes A., and Vanstone S. "Hand book of applied cryptography", The CRC-Press
series of Discrete Mathematics and its Applications CRC-Press,1997.
[20] Miyaji , Nakabayashi and Takano "Elliptic curves with low embedding degree", Journal of Cryptology, 2006, Volume 19, Number 4, Pages 553-562.