An Implementation Approach of Ecdlp-Based Diffie-Hellman Using Vb.Net

Dipti Aglawe
M.E Scholar,CSVTU
Dept of Computer Science Engineering
SSGI, Bhilai (C.G)

Samta Gajbhiye
Sr. Associate Professor CSVTU
Dept of Computer Science Engineering
SSGI, Bhilai (C.G)

Abstract

Elliptic curve cryptography [ECC] is a public-key cryptosystem like RSA. In wireless networks and mobile devices such as cell phone, PDA and smart card, security of wireless networks and mobile devices are prime concern. Elliptic curve can be applied to cryptography [5].

The principle attraction of ECC is that it appears to offer equal security for a far smaller key size, thereby reducing processor overhead. Applications that uses Discrete logarithm problem-based Diffie Hellman has more processing load as the key size has increased over recent years. However, the processing load is especially critical in a networks which have a relatively less bandwidth, slower CPU speed, limited battery power.

Elliptic curve Diffie-Hellman (ECDH) key exchange protocol is based on the elliptic curve discrete logarithm problem(ECDLP). This paper presents the implementation of the ECDLP-based Diffie-Hellman protocol for communication over insecure channel.

Key Terms: Elliptic curve cryptography, ECDLP-based Diffie-Hellman key exchange protocol.

I. Introduction

The fundamental goal of cryptography is to achieve privacy to enable two people to send each other messages over an insecure channel in such a way that only the intended recipient can read the message.

In a symmetric-key cryptographic system, the entities first agree upon the key. These keys should be secret and authentic. The major advantage of the symmetric-key cryptography is its high efficiency. However, there is a drawback of key distribution problem.

In 1976 Diffie-Hellman algorithm was presented by Whitfield Diffie and Martin E. Hellman[2] which solves the problem of key distribution. The Diffie-Hellman algorithm depends on the difficulty of computing discrete logarithms.

Key-exchange protocols are used in cryptography which two parties can communicate over an insecure network can generate a common secret key. Key exchange protocols are essential for enabling the use of shared-key cryptography to protect transmitted data over insecure networks.

.

Elliptic Curve Cryptography (ECC) is a public key cryptosystem based on elliptic curves. The basic advantage of using elliptic curves for cryptography purpose is that it appears to provide equal security for a far smaller key size, and thus reducing processing overhead[3].

The Diffie-Hellman key agreement protocol provide the secrecy of the shared key because only the communicating parties knows x_A and x_B, where x_A is the private key of user A and x_B is the private key of user B ,they can compute the shared secret key. The problem is that neither of the communicating parties is assured of the identity of the other, this problem can be solved if both parties have access to a trusted third party that issues certifications which assures their identity with the corresponding public key[1] .

It is important to mention that the Diffie-Hellman problem over elliptic curve with small keys is much harder to solve than the discrete logarithm over finite fields[2]

Rest of the paper is organized as follows. Section 2 describes the background which is necessary to understand the ECDLP-based Diffie-Hellman protocol. Section 3 shows the implementation of ECDLP-based Diffie Hellman algorithm in VB.NET. Section 4 is results .Finally, Section 5 is conclusion and future scope of the implementation.

Ii. Elliptic Curve Concept

An elliptic curve is a plane curve defined by an equation of the form

$y^2 = x^3 + ax + b$.   (1)

where x, y are elements of GF(p), and each value of the 'a' and 'b' gives a different elliptic curve.

In equation $y^2 = x^3 + ax + b$, a, b ∈ K and determinant$-16(4a^3 + 27b^2) \neq 0(\text{mod p})$. (2)

Here 'p' is known as modular prime integer making the EC finite field.

The condition that $-16(4a^3 + 27b^2) \neq 0$. implies that the curve has no "singular points", means that the polynomial $y^2 = x^3 + ax + b$ has distinct roots

The following shows the points of elliptic curve E(1,1) and p=29

$y^2 = x^3 + x + 1$

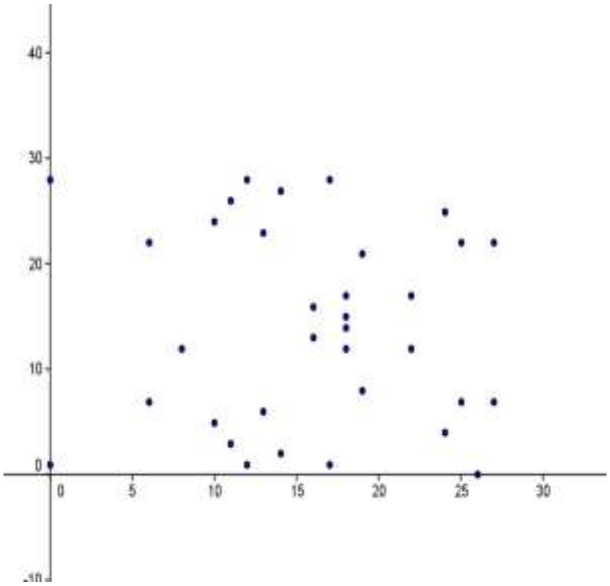


Fig1. Coordinate points of elliptic curve E(1,1) over E=29

An elliptic curve E consist of the solutions(x, y) defined by (1) and (2), along with an additional element called 0, which is the point of EC at infinity. The set of points (x, y) are said to be affine coordinate point representation.

The basic EC operations are point addition and point doubling.

Point addition **:** In order to find the sum of two points P and Q on elliptic curve E, we draw a line connecting P and Q. This line will intersect E at exactly one other point, which we will denote P * Q. P + Q will be defined as the reflection of P * Q across the x-axis.

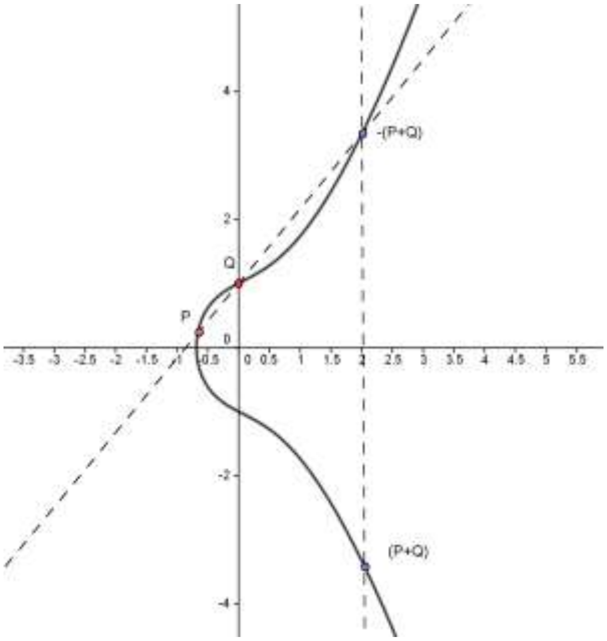


Fig2. Point addition operation

Point doubling: When P and Q are the same point, we draw the tangent line to E at P and find the second point where this line intersects E. We call this point P * P. Again, we reflect this point over the x-axis to obtain P + P.

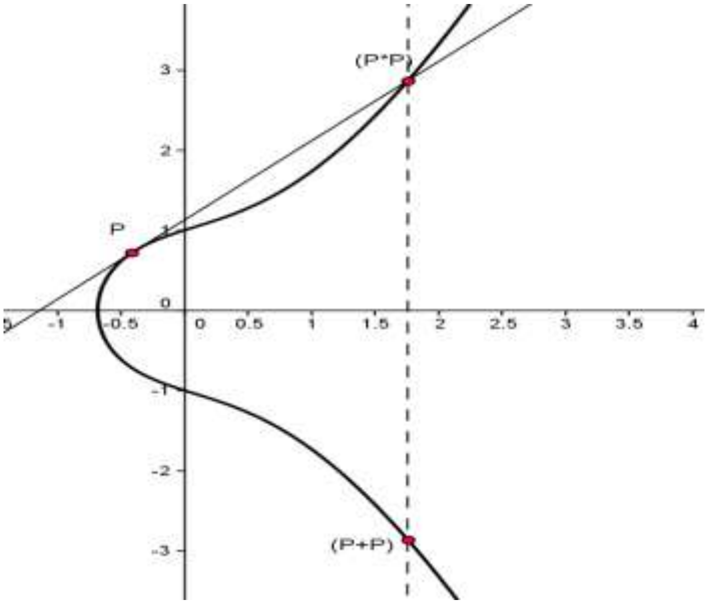


Fig3.Point doubling operation

The following figure shows case is where the line connecting P and Q is vertical. In this case, we define P + Q to be O, the point at infinity. Note that the line connecting any point and O will be a vertical line, and reflecting O about the x-axis results in O.

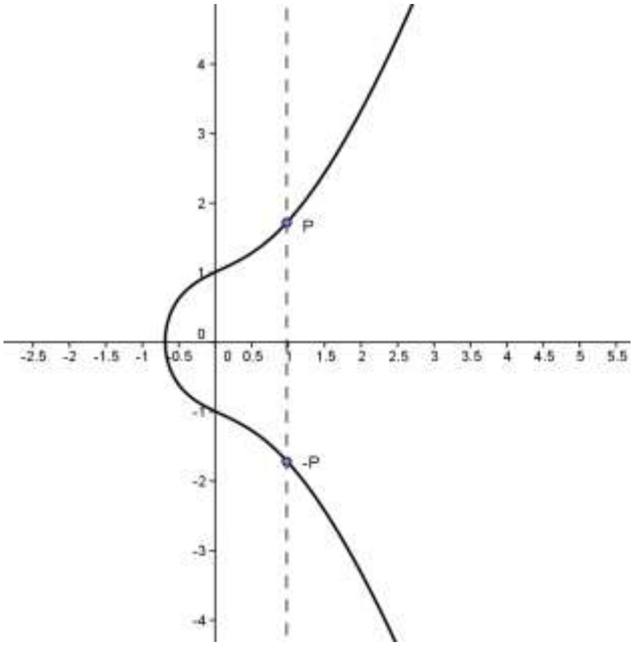


Fig4. Point at infinity

Let us start with P(xp,yp).
To determine 2P, P is doubled. This should be a point on EC. Use the following equation, which is a tangent to the curve at point P.

$S = [(3x_p{}^2 + a)/2y_p] \pmod p$ (3)

Then 2P has coordinates (x_r,y_r) given by:
$x_r = (S^2 - 2\, x_p) \bmod p$
$y_r = [S\,(x_p - x_r) - y_p] \pmod p$ (4)

For determining 3P, we use addition of points P and 2P, treating 2P=Q. Here P has coordinates $(x_{p,}y_p)$, Q=2P has coordinates (x_q,y_q). Now the slope is:

$S = [(y_q - y_p) / (x_q - x_p)] \bmod p$

P+Q=-R
$x_r = (S^2 - x_p - x_q) \bmod p$
$y_r = (S\,(x_p - x_r) - y_p) \bmod p$ (5)

The value of kP can be calculated by a series of doubling and addition operation

Iii. Implementation Of Ecdlp-Based Diffie Hellman Algorithm

Consider an elliptic curve over the field F_{29}, where the elliptic curve equation E: $y^2 = x^3 + ax + b$, we set a = 1 and b = 1, then we get the elliptic curve E: $y^2 = x^3 + x + 1$. This equation must satisfy the equation $4a^3 + 27b^2 \neq 0 \bmod p$ to form a group, this is verified. The following 36 points over E that satisfies this equation are[5]:

(0,1) (0,28) (6,7) (6,22) (8,12) (8,17) (10,5) (10,24) (11,3) (11,26) (12,1) (12,28) (13,6) (13,23) (14,2) (14,27) (16,13) (16,16) (17,1) (17,28) (18,14) (18,15) (19,8) (19,21) (22,12) (22,17) (24,4) (24,25) (25,7) (25,22) (26,0) (27,7) (27,22) (28,12) (28,17),O

These points may be graphed as below

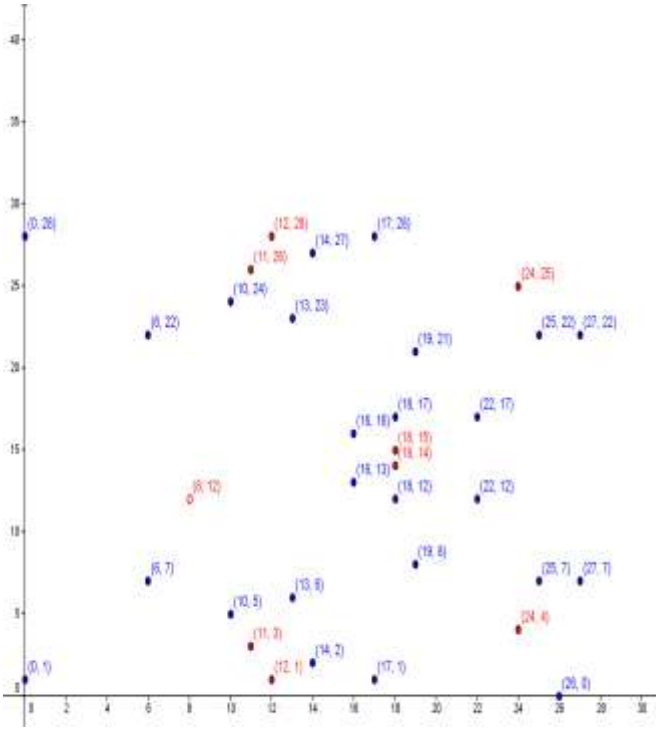


Fig5. Coordinate points for $E_{29}(1,1)$ with base points in red color

In our implementation ,we have used this curve for generation of shared key at both peer

Elliptic curve Diffie Hellman key exchange algorithm[3]:
1.A select an integer n_A less than n.This is user's A private key.
User A generate a public key $P_A = n_A\, G$;the public key is a point in Eq(a,b).
2.User B selects a private key n_B and computes the public key $P_B = n_B\, G$.

3. User A generates the secret key from B's public key
   $K=n_A\ n_B\ G$

4.Similarly, B generates the secret key from A's public key
   $K=\ n_B\ n_A G$.

The two calculations for the generation of secret key produces the same result.

IV. Results and Discussion

VB.NET socket layer programming has been used for connecting two systems. Messages are transmit from user A to user B when a socket is created.

In our implementation we use G=(11,3)[5] has been used as generator or base point.

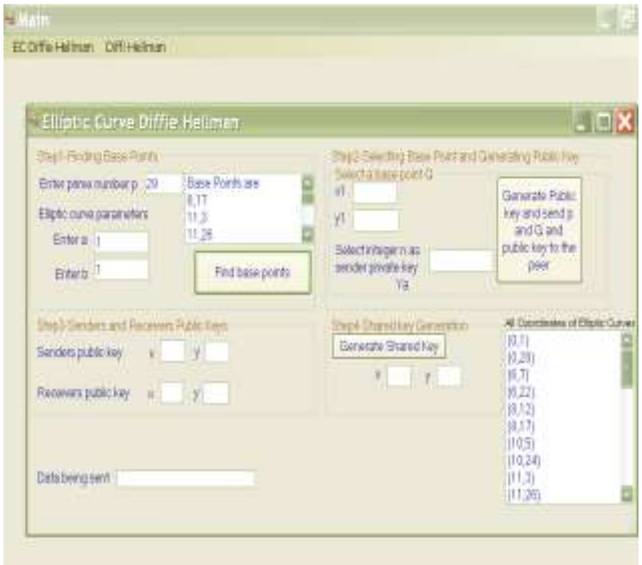
The following figure shows the communication between users A and user B

USER A	USER B
Step1:User A chooses a random number n_A=25	
Step2:User A computes public key $P_A= n_A G$=25*(22,3)=(24,4)	
	Step3: User B chooses a random secret n_B,=17
	Step4: User B compute public key $P_B = n_B\ G$ = 17(11,3) = (28,17) and sends to user A.
Step6: User A compute the shared secret $K=P_{AB}=n_An_BG$ =25*(12,17)=(12,28).	Step5:User B compute the shared secret $K=P_{AB}=n_Bn_AG$ = 17*(24,4)= (12,28).

Fig6: Steps to generate secret shared key

The following six GUI represents the above steps to generate secret shared key at both the ends.

Fig7 shows the parameters set by user A for elliptic curve with a=1,b=1 and p=29 and getting the coordinates and base points of the curve. In fig8 user A compute public key by selecting any base point and choosing a random number as private key. User A sends public key and parameter values to user B. User B gets all the parameters send by user A, and also the public key of user A as shown in fig9. Fig10 shows user B selecting private key and computing public key and send it to user A. Generation of shared key at user B is shown in fig11. Fig12 shows generation of shared key at user A.


Main
EC Diffie Hellman
Diffi Hellman
Elliptic Curve Diffie Hellman
Step1-Finding Base Points
Find base points
Step2-Selecting Base Point and Generating Public Key
Select a base point G
Generate Public key and send p and G and public key to the peer
Step3-Senders and Receivers Public Keys
Step4-Shared key Generation
Generate Shared Key
All Coordinates of Elliptic Curves

Enter prime number p 29

Elliptic curve parameters

Enter a 1

Enter b 1

Base Points are
8,17
11,3
11,26

x1

y1

Select integer n as sender private key Ya

Senders public key x y

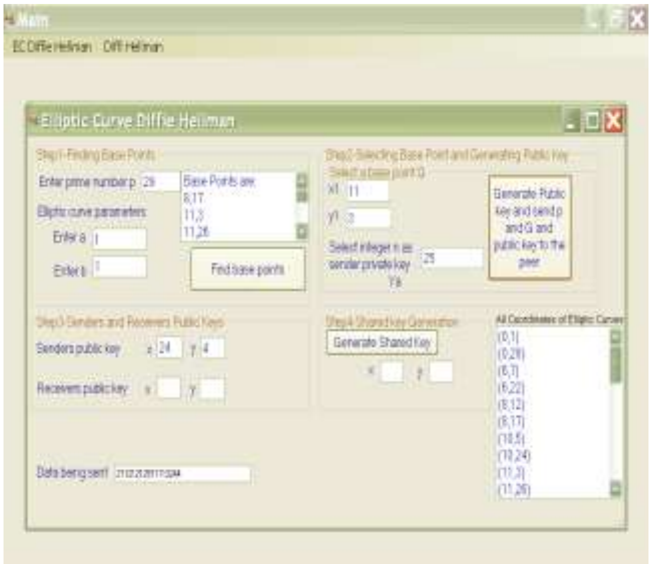
Receivers public key x y

x y

Data being sent

(0,1)
(0,28)
(6,7)
(6,22)
(8,12)
(8,17)
(10,5)
(10,24)
(11,3)
(11,26)

Fig7: Displays the coordinate and base points of elliptic curve


Main
EC Diffie Hellman
Diffi Hellman
Elliptic Curve Diffie Hellman
Step1-Finding Base Points
Find base points
Step2-Selecting Base Point and Generating Public Key
Select a base point G
Generate Public key and send p and G and public key to the peer
Step3-Senders and Receivers Public Keys
Step4-Shared key Generation
Generate Shared Key
All Coordinates of Elliptic Curves

Enter prime number p 29

Elliptic curve parameters

Enter a 1

Enter b 1

Base Points are
8,17
11,3
11,26

X1 11

y1 3

Select integer n as sender private key Ya 25

Senders public key x 24 y 4

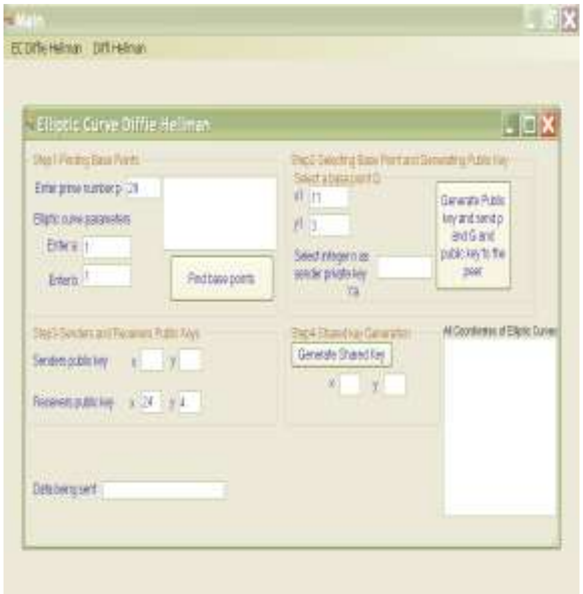
Receivers public key x y

x y

Data being sent [illegible]

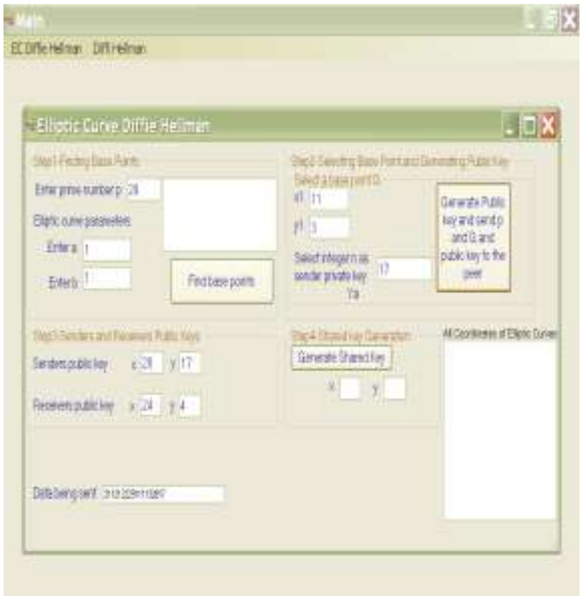
(0,1)
(0,28)
(6,7)
(6,22)
(8,12)
(8,17)
(10,5)
(10,24)
(11,3)
(11,26)

Fig8: Shows user A compute public key and sending to user B


Main
EC Diffie Hellman
Diffi Hellman
Elliptic Curve Diffie Hellman
Step1-Finding Base Points
Find base points
Step2-Selecting Base Point and Generating Public Key
Select a base point G
Generate Public key and send p and G and public key to the peer
Step3-Senders and Receivers Public Keys
Step4-Shared key Generation
Generate Shared Key
All Coordinates of Elliptic Curve

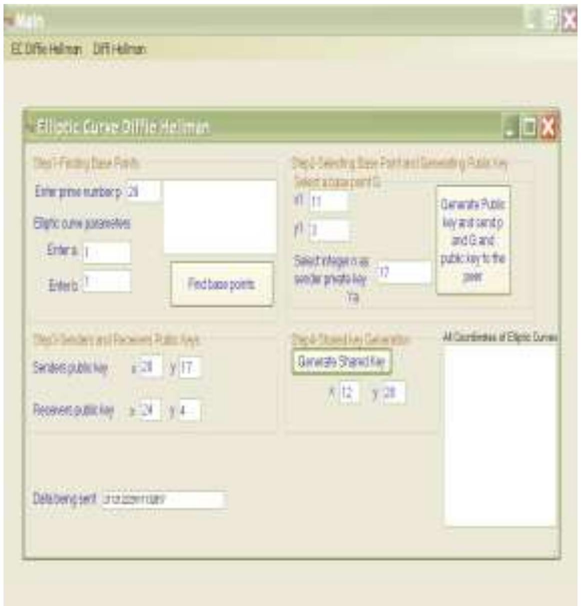
Enter prime number p: 29
Elliptic curve parameters
Enter a: 1
Enter b: 1
x1: 11
y1: 3
Select integer n as sender private key Ya:
Senders public key x: y:
Receivers public key x: 24 y: 4
Data being sent:

Fig9:Shows user B receives public key of user A


Main
EC Diffie Hellman
Diffi Hellman
Elliptic Curve Diffie Hellman
Step1-Finding Base Points
Find base points
Step2-Selecting Base Point and Generating Public Key
Select a base point G
Generate Public key and send p and G and public key to the peer
Step3-Senders and Receivers Public Keys
Step4-Shared key Generation
Generate Shared Key
All Coordinates of Elliptic Curve

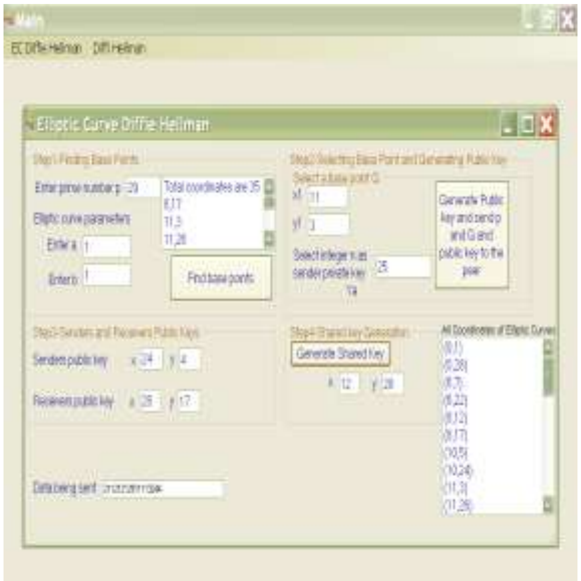
Enter prime number p: 29
Elliptic curve parameters
Enter a: 1
Enter b: 1
x1: 11
y1: 3
Select integer n as sender private key Ya: 17
Senders public key x: 28 y: 17
Receivers public key x: 24 y: 4
Data being sent: [illegible]

Fig10: Shows user B compute its public key and send it to user A


Main
EC Diffie Hellman
Diffi Hellman
Elliptic Curve Diffie Hellman
Step1-Finding Base Points
Find base points
Step2-Selecting Base Point and Generating Public Key
Select a base point G
Generate Public key and send p and G and public key to the peer
Step3-Senders and Receivers Public Keys
Step4-Shared key Generation
Generate Shared Key
All Coordinates of Elliptic Curve

Enter prime number p: 29
Elliptic curve parameters
Enter a: 1
Enter b: 1
x1: 11
y1: 3
Select integer n as sender private key Ya: 17
Senders public key x: 28 y: 17
Receivers public key x: 24 y: 4
Shared key x: 12 y: 28
Data being sent: [illegible]

Fig11: Displays shared key at user B.


Main
EC Diffie Hellman
Diffi Hellman
Elliptic Curve Diffie Hellman
Step1-Finding Base Points
Find base points
Step2-Selecting Base Point and Generating Public Key
Select a base point G
Generate Public key and send p and G and public key to the peer
Step3-Senders and Receivers Public Keys
Step4-Shared key Generation
Generate Shared Key
All Coordinates of Elliptic Curve

Enter prime number p: 29
Total coordinates are 35
8,17
11,3
11,26
Elliptic curve parameters
Enter a: 1
Enter b: 1
x1: 11
y1: 3
Select integer n as sender private key Ya: 25
Senders public key x: 24 y: 4
Receivers public key x: 28 y: 17
Shared key x: 12 y: 28
Data being sent: [illegible]
All Coordinates of Elliptic Curve:
(0,1)
(0,28)
(8,7)
(8,22)
(9,12)
(9,17)
(10,5)
(10,24)
(11,3)
(11,26)

Fig12: Displays shared key at user A

V. Conclusion and Future Scope

The secret key generated by ECDLP-based Diffie Hellman is a pair of numbers. This key can be used as a session key for encryption or decryption of messages by using any conventional symmetric key algorithm.

Elliptic curve groups are based on scalar multiplication. Consider the equation $P_2=kP_1$ where P_1 and P_2 are the elliptic curve coordinates and k< p. It is easy to calculate P_2 given k and p(prime number), but it is relatively hard to determine k given P_2 and P_1.This is called elliptic curve discrete logarithm problem(ECDLP).

Consider the group $E_{29}(1,1)$ with P_1= (11,3) and P_2=(24,4).The brute-force method is to compute multiples of P_1until P_2 is found. Thus $25P_1$=(24,4) i.e k=25.In real application, k would be so large as to make the brute-force approach to take time in months or years.

Diffie-Hellman over elliptic curve is implemented on many small devices (e.g. smart card) where limited processing power and limited memory capacity exist, this is due to the small number of bits required to perform the encryption and decryption process. Elliptic curves are considered newly in cryptography and is one of the most researched topic in cryptography

ECDLP-based Diffie-hellman algorithm can be used in applications where security is required. Mobile computing, wireless networks, server based encryption, encryption for images, financial communication protocols, military based applications and many other. There is a lot of research required for its practical implementation.

References:

[1] Malek Jakob Kakish, “A secure Diffie-Hellman schemes over elliptic curves”, IJRRAS, vol.10, no 1, pp 98- 106, 2012.

[2] W. Diffie and M. E. Hellman, “New directions in cryptography,” IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644–654, November 1976.

[3] Williams Stallings, Cryptography and Network Security, Prentice Hall, 4th Edition, 2006

[4] Dr H.K.Pathak,” Discrete mathematical structure”

[5] Ms Dipti Aglawe and Prof Samta Gajbhiye, “Software implementation of cyclic abelian elliptic curve using matlab”, International Journal of Computer Applications (0975 – 8887) Volume 42– No.6, March 2012