IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 4, Ver. I (Jul-Aug. 2014), PP 65-68
www.iosrjournals.org

Implementation of Ecc Using Socket Programming In Java.

[1]Ms. Shubhi Gupta, [2]Ms. Swati Vashisht
[1,2]Assistant Professor, Department of Computer Science and Engineering,DIT School of Engineering, Greater Noida – 201308, India

Abstract: *Elliptic Curve Cryptography (ECC) has attracted the attention of researchers and developers as it has got robust mathematical structure and higher security compared to algorithms like RSA (Rivest Adleman and Shameer Public key Algorithm). It increases the security compared to RSA or at least give comparable security with lesser key size than RSA. This research shows the implementation of ECC algorithm on a data that is sent from one computer device to another connected through LAN using socket programming in Java.*
Index terms: *ECC, Socket programming, ASCII*

I. Introduction

To secure the information secure, a science called as Cryptography is used. It is a branch of applied mathematics that emphasizes on adding security on any type of message. Cryptographic algorithm has encryption keys that change a general algorithm into a method of encryption. This 'key' is the crux or seed of this algorithm.

The Elliptic curves in Cryptography were generated by Victor Miller and N. Koblitz. It involves factorization or discrete log problem in sub-exponential time. This means smaller parameters can be used with smaller key size to perform faster computations.

ECC Cryptographic System:

In this type of Public key cryptography, the user or the communicating device should have a pair of key, public key and a private key. To carry the encryption and decryption process, some set of operations are performed on these keys. The underlying mathematic operation is defined over the elliptic curve $y^2 = (x^3 + ax + b) \bmod p$ such that $4a^3 + 27b^2 \bmod p \neq 0$ where p is a large prime number and a and b are the coefficients that generates different elliptic curve points (x,y).

Operation:

1. Take a large prime no. p and values for coefficients a and b such that $4a^3 + 27b^2 \bmod p \neq 0$.
2. Consider an equation: $y^2 = (x^3 + ax + b) \bmod p$.
3. Take all values of y between 0 to p-1 and calculate $y^2 \bmod p$.
4. Take all values of x between 0 to p-1 and calculate $(x^3 + ax + b) \bmod p$.
5. Collect values of y from step 3 corresponding to values computed in step 4.
6. Collect all points (x,y) from step 5.
7. Input a supposed value of G known here as the base point which belongs to points from step 6.
8. Calculate 2G, 3G… such that :

2G = G + G, 3G = 2G + G and so on until a value iG is found (let i be the least positive integer) such that the value of x coordinate of this point is same as the value of x coordinate of G and the value of y coordinate is prime number minus the value of y coordinate of G. From this, Order of G, called as n is computed as i+1.

For instance, If p = 7, G = (1,3) , values of 2G, 3G etc. will be calculated until the value of 3G comes out to be (1,4). Then the order, n will be 4.

This addition will be done as follows:

If point R = point P + point Q

New point (x_R, y_R) : $x_R = (\lambda^2 - x_P - x_Q) \bmod p$

$y_R = ((\lambda (x_P - x_R) - y_P) \bmod p)$

where $\lambda = (y_Q - y_P) / (x_Q - x_P) \bmod p$ if $P = Q$ and

$(3x_P^2 + a) / 2y_P) \bmod p$ if $P \neq Q$

9. Computation of sender's (say, A) public key:

Choose a large integer n_A, so that it lies between 1 and n.

Compute $P_A = n_A G$

10. Computation of receiver's (say, B) public key:

Choose a large integer n_B, so that it lies between 1 and n.

Compute $P_B = n_B G$

11. Encryption:

A will encrypt the message with B's public key –

Let plain text P_m belongs to the point set in computed in step 6.

Let k be the random integer which lies between 1 and n.

Compute – $(kG, P_m + kP_B)$

12. Decryption :

Compute kGn_B.

Compute $P_m + k P_B - kGn_B$ to get P_m.

Socket Programming in Java:

Sockets provides the communication contrivance between two computer systems using Transmission Control Protocol i.e. TCP. A socket on its end of communication is created by a client program which then attempts to connect itself to a server. The server creates a socket object on its end of the communication only when the connection is made. The client and server are then ready to communicate by writing to and reading from the socket.

A socket is represented by java.net.Socket class the java.net.ServerSocket class acts as a framework for the server program to establish connections with clients upon receiving such requests.

To establish a TCP connection between two systems using sockets:

- A ServerSocket object is instantiated by a server, denoting the port number through which the communication is to occur on.
- The accept() method of the ServerSocket class is invoked by the server. This method waits for a client to connect to the server on the given port.
- When the server is waiting, a client initiates a Socket object, specifying the server name and port number to which it has to connect.
- The connection between the client, specified server and the port number is attempted by the constructor of the socket class. If it gets successful in establishing the connection, the client now has a Socket object which is capable of communicating with the server.
- The accept() method reciprocates with a reference to a new socket on the server connected to the client's socket.

Once the connections are established, I/O streams enable the communication. Each socket has an OutputStream as well as an InputStream. The client's OutputStream is linked to the server's InputStream, and the client's InputStream is linked to the server's OutputStream.

II. Software Implementation

This implements ECC using a data string which can be alphanumeric which is carried on two different computer systems using socket programming in Java. Socket programming enabled entering the string at one computer system, encrypting it and transmitting it to the other. The receiver's computer system can decrypt it. The encryption and decryption of a particular character is done by using its ASCII value.

For instance: a = 0,
b = -4,
Base point, G = (31, 6),
Sender's private key = 25,
Receiver's private key = 35,
Random key = 67.
This created over 200 different coordinates of an elliptic curve. The plaintext then becomes the coordinate stored at the number denoting the corresponding ASCII value.
The whole encryption and decryption process is processed through various code modules:

- EC Point: This spawns the coordinates of an elliptic curve.
- Socket Client: This percolates the string on client system, encrypts it and sends it to the server system.
- Socket Server: This procures the encrypted string, decrypts it to plain text string.
- **Encryption**: A character is picked as plaintext. The ASCII value corresponding to it is taken into as an integer variable. The point on the elliptic curve corresponding to this particular integer is selected from the database. This following point is then encrypted. Now, this resultant point is mapped again to the database that will correspond to a new integer value. The new integer is then changed to a corresponding character which will consist of two specifications - printable ASCII character which further acts as an index and page number to which the corresponding index belongs to.
- **Decryption:** It selects the encrypted character and the coinciding page number. This calculates back the integer. Reverse mapping is carried out for the conversion of integer to point. Decryption is carried out. This again helps in getting integer from the database. The corresponding character is the plain text character.
- The printable ASCII character ranges from 32 to 126 only. If the encryption character goes beyond this range, additional calculation is done. A tilde (~) is sent and the ASCII value gets incremented by 32 to send as a printable character whereas on the decryption side, reverse calculation is done when tilde is detected.

Plain Text: It works!!
Encrypted String: /~-~qI+RA##
Decrypted String: It works!!

III. Conclusion

This paper gives a brief illustration of ECC key exchange and encryption/decryption. The implementation of ECC on text document using socket programming in java is explained in detail. This approach is in accordance to all security aspects of elliptic curves and applies to all ASCII characters.

References

[1]. William Stallings, Cryptography and network security, 2nd edition, Prentice Hall publications

[2]. B.Schiener, Applied Cryptography. John Wiley publications and sons, 2nd edition, 1996

[3]. Victor Miller, "Uses of elliptic curves in cryptography", Advances in cryptology, 1986

[4]. N.Koblitz, A course in number theory and cryptography.

[5]. http://www.tutorialspoint.com/java/java_networking.htm

[6]. Kohlekar Megha, Jadhav Anita, 2011." Implementation of Elliptic Curve Cryptography on Text and Image", International Journal of Enterprise Computing and Business Systems, Vol. 1 Issue 2 July 2011.