SECURING THROUGH PSEUDORANDOM NUMBER GENERATOR AND HASHING IN CRYPTOGRAPHY: REVIEW

[1]Aatif Jamshed, [2]Megha Bhardwaj, [3]Medhavi Pandey &[4]Dr. Krishna Kant Agrawal
[1,2,3] Assistant Professor, [4]Head of Department
[1,2,3,4] Computer Science& Engineering
[1,2,3,4] Delhi Technical Campus, Greater Noida, India

Abstract: Authentication and confidentiality are two major aspect of the cryptography. In no way you can say that your message is completely secure i.e. any third party can attack you message at any time/stage of processing. For sending confidential messages the security feature is essential to avoid any unauthorized access. We have to encode a message before encryption in ECC as describe by Sir Koblitz. In this paper we employed a new mechanism to secure the value of k by using PRN generator (because of confusion and diffusion) and the value of m by using SHA (for the use of one-way property). Based on the simulated results we achieve strong mechanism of authentication and confidentiality that ultimately enhance the data security.

Index Terms: **Encoding, Decoding, ECC, PRN, Hash function, Point encryption.**

I. INTRODUCTION:

Elliptic curve cryptography (ECC) was discovered in 1985 by Neal Koblitz and Victor Miller. Elliptic curve cryptographic schemes are public-key mechanisms that provide the same functionality as RSA schemes. However, their security is based on the hardness of a different problem, namely the elliptic curve discrete logarithm problem (ECDLP). Currently the best algorithms known to solve the ECDLP have fully exponential running time, in contrast to the sub exponential-time algorithms known for the integer factorization problem. This means that a desired security level can be attained with significantly smaller keys in elliptic curve systems than is possible with their RSA counterparts.

The advantages that can be gained from smaller key sizes include speed and efficient use of power, bandwidth, and storage. Domain Parameters [3] are the set of predefined constants to be known by all the devices in ECC. A pseudorandom number generator (PRNG)[12], also known as a deterministic random bit generator (DRBG) is an algorithm for generating a sequence of numbers that approximates the properties of random numbers. The sequence is not truly random in that it is completely determined by a relatively small set of initial values, called the PRNG's state, which may include a truly random seed.

The Secure Hash Algorithm[11] specifies four algorithms - SHA-1, SHA-256, SHA-384, and SHA-512 - for computing a condensed representation of electronic data/message. When a message of any length < 264 bits (for SHA-1 and SHA-256) or < 2128 bits (for SHA-384 and SHA-512) is input to an algorithm, the result is an output called a message digest. The message digests range in length from 160 to 512 bits, depending on the algorithm. Secure hash algorithms are typically used with other cryptographic algorithms, such as digital signature algorithms and keyed-hash message authentication codes, or in the generation of random numbers (bits).

The general cubic equation of elliptic curves is

$$y^2+axy+by=x^3+cx^2+dx+e. \qquad (1.1)$$

By not going into that much in detail let we convert our message in points by restricting our equation to the form

$$y^2 = x^3 + ax + b \qquad (1.2)$$

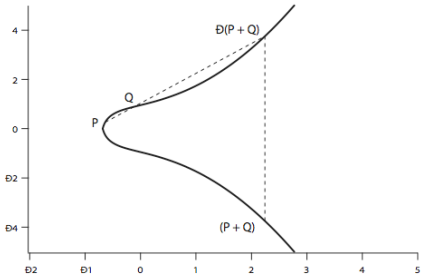

Fig.1 shows a general curve for the above equation.

Say E_P (a, b) consisting of all the points (x, y) that satisfy the above equation together with element at infinity O. A group can be defined based on the set E_P (a, b) for specific values of a and b. If P, Q, R are points on E_P (a, b) the relations commutativity, associativity, existence of an identity element and existence of inverse hold good. The heart of ECC is discrete logarithm problem [4] that can be stated as "it should be very hard to find a value k such that **Q=kP** where P and Q are known'. But 'it should be relatively easy to find Q where k and P are known' P, Q are points on the elliptic curve.

II. ELLIPTIC CURVE EXAMPLE:

Step1: Consider the equation of the curve as $y^2 \bmod p = (x^3 + ax + b) \bmod p$
Step2: Take the inputs a, b, p (p is key of the ECC algorithm)
Step3: Choose two non-negative integers a, b and a large prime number such that $4a^3 + 27b^2 \bmod p \neq 0$.

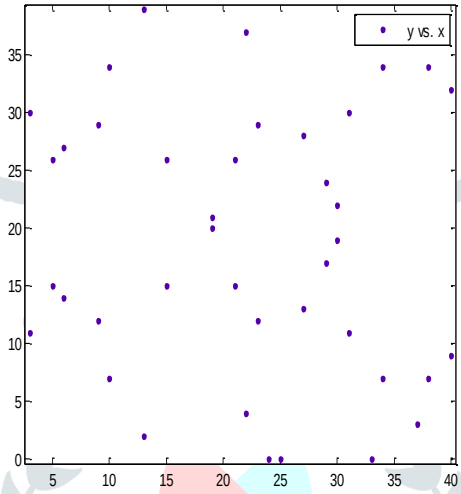

Fig.2 shows the elliptic curve, $y^2 \bmod 41 = (x^3 + 3x + 3) \bmod 41$.

We choose the parameter as-
a= 3
b= 3
p= 41

The set of points on the above curve are
{
[3 11], [3 30], [5 15], [5 26], [6 14], [6 27], [9 12], [9 29], [10 7], [10 34], [13 2], [13 39], [15 15], [15 26], [19 20], [19 21], [21 15], [21 26], [22 4], [22 37], [23 12], [23 29], [24 0], [25 0], [27 13], [27 28], [29 17], [29 24], [30 19], [30 22], [31 11], [31 30], [33 0], [34 7], [34 34], [37 3], [37 38], [38 7], [38 34], [40 9], [40 32]
}

Multiplication of a point with a positive integer k is defined as the sum of copies of P, k times. This operation is called Point Multiplication in ECC. So 3P=P+P+P.
The above points form the Group i.e. Ep(a,b). Each X and Y coordinate ranges between 0 and 40. The addition of the two points on the curve and the inverse of a point on the curve are defined in the field using modular arithmetic. The point at the infinity is identity point on the curve.

III. ECC PUBLIC KEY CRYPTOSYSTEM

In the public key elliptic curve cryptosystems, we assume that entity A wants to send a message m to entity B securely. Order of a point on the curve can be defined as a value n such that

$$nP = P+P+..+P..\ n \text{ times} = O \text{ (infinity)} \qquad (3.1)$$

3.1. Key generation:

Both the entities in the cryptosystem agree upon a,b,p,G,n which are called 'Domain Parameters' of ECC. G is called generator point and n is the order of G.
Now A generates a random number $n_A < n$ as his private Key and calculates his public key Set $P_A = G+G+G\ldots+n_A$ times.
B generates a random number $n_B < n$ as his private Key and calculates his public key, set $P_B = G+G+G\ldots+n_B$ times.

3.2. Key Exchange:

Entity A computes his Shared Key by Computing $K = P_A + P_A + ..+\ldots\ n_B$ times
Entity A computes his Shared Key by Computing $K = P_B + P_B + ..+\ldots\ n_A$ times

The two above keys have same value because:

$$n_A * P_B = n_A * (n_B * G) = n_B * (n_A * G) = n_B * P_A \qquad (3.2)$$

3.3. Encryption:

A sends Cm = 2 ciphertext points those are { kG, Pm + k PB)}.
Where G - generator Point
Pm - plaintext point on the curve
k - a random number chosen by A
PB - public key of B

3.4. Decryption:

$$Pm + kPB - nB(kG) = Pm + k(nB)G - nB(kG) = Pm \qquad (3.3)$$

3.5. Method of Encoding:

We have a systematic way of finding points on Ep(a,b) relating somehow to the plaintext message. Select a curve on which we will get a minimum of 256 points, so that we fix each point on the curve by the value given to list. For example, 'SECURITY' can be written as sequence of characters that is '83' '69' '67' '85' '82' '73' '84' '89' we can map these values to fixed points on the curve. This is easiest method for embedding a message but less efficient in terms of security.

IV. PROPOSED METHOD FOR ENCODING PLAINTEXT [6]:

Step1: Pick an elliptic curve Ep(a,b).
Step 2: Let us say that E has N points on it.
Step 3: Let us say that our alphabet consists the digits as generated by ASCII coding.
Step 4: This converts our message into a series of numbers between 0 and 126.
Step 5: Now choose an auxiliary base parameter, which should be generated by a Pseudo Random Number Generator. (both parties should agree upon this)
Step 6: Use SHA while transferring the value of m.
Step 7: Calculate all possible pair of x and y, respectively check for x=(*mk*)*mod p*, then $x=(mk + 1) \bmod p$ … and try to solve for *y*.
Step 8: In practice, we will find such a *y* before we hit $x = (mk + k - 1) \bmod p$. Then take the point (x,y). This now converts the number m into a point on the elliptic curve. In this way, the entire message becomes a sequence of points.

V. DECODING:

Consider each point (*x*,*y*) and set *m* to be the greatest integer less than (*x*-1)/*k*. Then the point (*x*,*y*) decodes as the symbol *m*.

VI. CONCLUSIONS:

The Execution time for encoding and decoding functions is increased slightly in comparison of original method. Uses of compression technique enhance the efficiency and also make the message more secure. The use of PRN and SHA make the method much more secure than other cryptographic method.

VII. REFERENCES:

[1] Darrel Hankerson, Julio Lopez Hernandez, Alfred Menezes, *Software Implementation of Elliptic Curve Cryptography over Binary Fields*, 2000,Available at http://citeseer.ist.psu.edu/hankerson00software.html
[2] Certicom, Standards for Efficient Cryptography, *SEC 1: Elliptic Curve Cryptography, Version 1.0*, September 2000, Available athttp://www.secg.org/download/aid-385/sec1_final.pdf
[3] Certicom, Standards for Efficient Cryptography, *SEC 2: Recommended Elliptic Curve Domain Parameters, Version 1.0*, September 2000,Available athttp://www.secg.org/download/aid-386/sec2_final.pdf
[4] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
[5] William Stallings, *Cryptography and Network Security, Principles and Practice.* ed.,Prentice Hall, New Jersey,2003.
[6]. N. Koblitz, \Elliptic curve cryptosystems", *Mathematics of Computation*, 48 (1987), 203-209.
[7] MATLAB Summary and Tutorial, *www.math.ufl.edu/help/matlabtutorial*
[8] R. Schoof. Elliptic Curves Over Finite Fields and the Computation of Square Roots mod p. Mathematics of Computation, Vol. 44, No. 170, pp. 483-494, April 85.
[9] F. Morain. Building cyclic elliptic curves modulo large primes. Advances in Cryptology - EUROCRYPT '91, Lecture Notes in Computer Science, 547: 328-336,1991.
[10]Chandhok, S., Anand, R., Gupta, S., & Jamshed, A. (2017). An Analysis of Sentimental Data using Machine Learning Techniques. International Journal of Computer Applications, 166(3).
[11] N. Koblitz. *A Course in Number Theory and Cryptography*, Springer- Verlag, second edition, 1994. ISSN

[12]Federal Information-Processing Standards Publication 180-2-2002 August 1-Announcing the-SECURE HASH STANDARD
[13]*en.wikipedia.org/wiki/**Pseudorandom_number_generator***