Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 13, Number 9 (2017), pp. 4995-5011
© Research India Publications
http://www.ripublication.com

The Advantages of Elliptic Curve Cryptography for Security

Dr. S. Vasundhara
Asst Professor of Mathematics
G.Narayanamma Institute of Technology & Science(women)
Shaikpet. Hyderaabad, India.

Abstract

Elliptic Curve Cryptography has been a recent research area in the field of Cryptography. It provides higher level of security with lesser key size compared to other Cryptographic techniques. This paper provides an overview of ellipticcurves and their use in cryptography. The focus is on the performance advantages to be obtained by using elliptic curve cryptography instead of a traditional cryptosystem like RSA. Specific applications to secure messaging and identity-based encryption are discussed.

Keywords: Cryptography; Elliptic curve cryptography; Point addition; Point doubling

1. INTRODUCTION

Cryptography is transformation of plain message to make them secure and immune from intruders. Elliptic Curve Cryptography (ECC) is a public key cryptography developed independently by Victor Miller and Neal Koblitz in the year 1985. In Elliptic Curve Cryptography we will be using the curve equation of the form

$$Y^2 = x^3 + ax + b \qquad (1)$$

which is known as Weierstrass equation, where a and b are the constant with

$$4a^3 + 27b^2 \neq 0 \qquad (2)$$

1.1 Mathematics in elliptic curve cryptography over finite field

Cryptographic operation on elliptic curve over finite field are done using the coordinate points of the elliptic curve.

Elliptic curve over finite field equation is given by:

$$y2 = \{x3 + ax + b\} \bmod\{p\} \qquad (3)$$

Certain formula are defined for operation with the points.

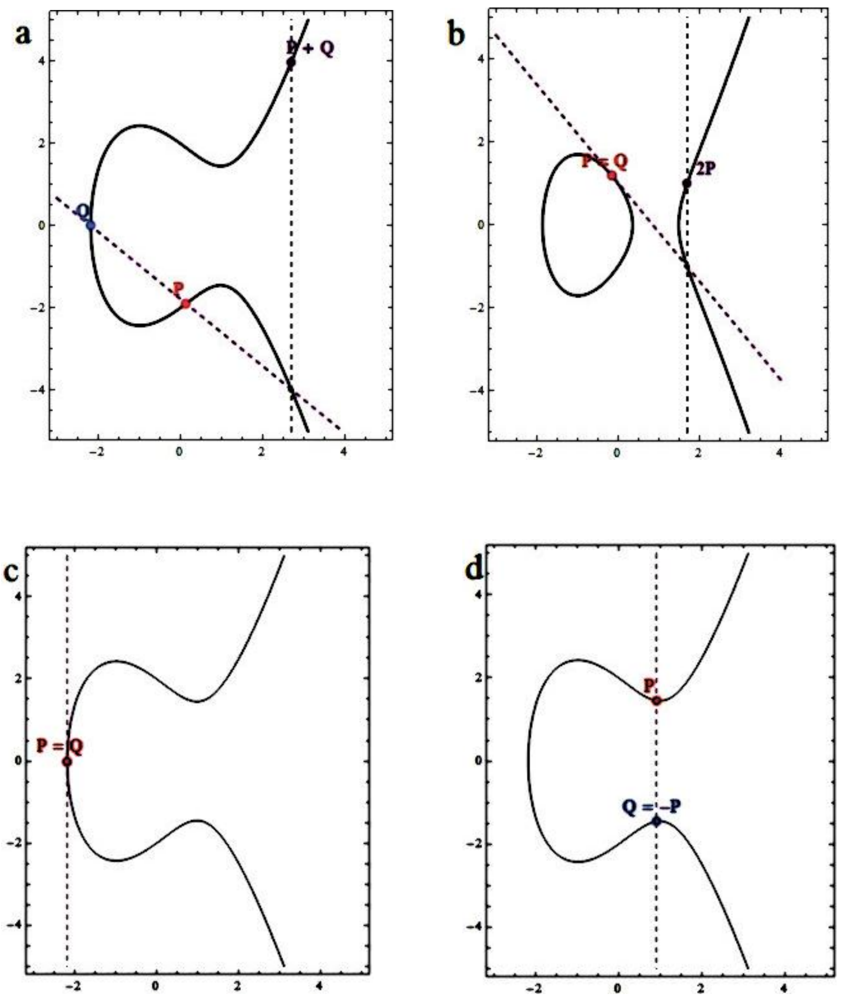


Fig. 1. (a) Point addition; (b) Point doubling; (c) Point at infinity when *y* coordinates are both 0; (d) Point at infinity when the coordinates are mirror image of each other.

1.1.1 Point addition

The two point *P(x*1, *y*1*)* and *Q(x*2, *y*2*)* are distinct. *P* + *Q* = *R(x*3, *y*3*)* is given by the following calculation.

Figure 1(a) shows graphical representation of Point Addition operation.

$\lambda = (Y_2 - Y_1) / (X_2 - X_1)$
$X_3 = \lambda^2 - X_1 - X_2$ and

$Y_3 = -Y_1 + \lambda (X_1 - X_3)$

1.1.2 Point Doubling

The two point *P(x*1, *y*1*)* and *Q(x*1, *y*1*)* overlap. *P* + *Q* = *R(x*3, *y*3*)* is given by the following calculation.

Figure 1(b) shows graphical representation of Point Doubling operation.

$(X_1, Y_1) + (X_2, Y_2) = (X_3, Y_3)$;

Where $Y_1 \neq 0$
2P = R where
$\lambda = (3X_1^2 + a) / (2Y_1)$
$X_3 = \lambda^2 - 2X_1$ and

$Y_3 = \lambda (X_1 - X_3) - Y_1$

1.1.3 Point multiplication

Let *P* be any point on the elliptic curve. Multiplication operation over *P* is defined by the repeated addition.

$kP = P + P + P + \cdot\cdot\cdot + k$ times.

1.1.4 Point at infinity

If $x_1 = x_2$ and$y_1 = y_2 = 0$ or $x_1 = x_2$ and $y_1 = -y_2$, the points is said to intersect at infinity denoted by *O*.

Figure 1(c) and 1(d) shows graphical representation of Point at Infinity

1.1.5 Finding Inverse Modulo

Let us consider an elliptic curve

$Y^2 = x^3 + 2x + 4 \bmod 7$

It has got the following coordinate points. {O, {0, 2}, {0, 5}, {1, 0}, {2, 3}, {2, 4}, {3, 3}, {3, 4}, {6, 1}, {6, 6}}.

To perform point addition of two points {0, 5} and {3, 4}, we need to find lambda.

$\lambda = \frac{4-5}{3-0}\ mod 7$

then

$\lambda = \frac{-1}{3} \text{mod} 7$

For finding Inverse Modulo we have used Extended Euclidean Algorithm shown in Table 1.

Table 1. Finding inverse modulo using extended euclidean algorithm.

Q	R_1	R_2	R	T_1	T_2	T
2	7	3	1	0	1	-2
3	3	1	0	1	-2	7

Where

• Q = Quotient for R_1 divided by R_2;

• R_1 = Modulus value initially, followed by left shift of previous value of R_2 in later cases.

• R_2 = Denominator value initially, followed by left shift of value from previous R in later cases.

• R = Remainder of R_1 divided by R_2.

• T_1 = 0 initially, followed by left shift of previous value from T_2.

• T_2 = 1 initially, followed by right shift of previous value from T .

• $T = T_1 - Q * T_2$.

• Continue till $R = 0$, and inverse modulo is given by the value at T_2.

$\lambda = -1 * -2 \text{mod } 7$

$\lambda = 2$

1.2 Elliptic curve cryptography

Since ECC is a public key cryptography, we require a public key and a private key. Consider Alice and Bob are thetwo communicating parties. They agree upon a common Elliptic curve equation and a generator G. Let Alice and Bobprivate keys be nA and nB respectively. Alice and Bob public keys are given by

$$Pa = nAG$$

and

$$Pb = nBG$$

respectively. If Alice want to send a message 'Pm' to Bob, Alice uses Bob's public key to encrypt the message. Thecipher text is given by

$$Pc = \{kG, Pm + kPb\}$$

where 'k' is a random integer. The random 'k' make sure that even for a same message the cipher text generated is different each time. This gives a hard time for someone who is illegally trying to decrypt the message. Bob decrypts the message by subtracting the coordinate of 'kG' multiplied by nB from '$Pm + kPb$'.

$$Pm = \{Pm + kPb - nBkG\}$$

Here multiplied does not mean simple multiplication that we do in algebra, rather it is multiple addition of points using the point addition method stated above in point multiplication. As the multiplier nB is the secret key of Bob, only Bob can decrypt the message sent by Alice.

2. LITERATURE REVIEW

Many authors have exploited the strength of ECC and came up with implementation in various tasks of public key cryptography like authentication, digital signature, key agreement and encryption. Victor S. Miller explain the use of Elliptic curves in Cryptography. He proposed an encryption scheme similar to Diffie-Hellman key exchangeprotocol but faster by around 20 percent. Neal Koblitz explain about the elliptic curves over finite fields for public key cryptosystems. He explain that the discrete logarithm problem is harder for finite group field compared to binary field. He also gave a theorem for nonsmoothness existing in cyclic subgroup generated by a global point. Neal Koblitz, Alfred Menezes and Scott Vanstone extended the idea of discrete logarithmic problem used in public key cryptography of Diffie-Hellman to elliptic curve group. It provided smaller block size, high speed and high security. Darrel Hankerson, Alfred Menezes and Scott Vanstone wrote a book called Guide to Elliptic Curve Cryptography and it provide various details of elliptic curve arithmetic, cryptographic protocols and implementation issues. Lawrence C. Washington wrote a book called Elliptic Curves: Number Theory and Cryptography. It provides proofs to many theorem to understand elliptic curves. Jorko Teeriaho gave a very clear example implementation of ECC-DH key exchange, ECC encryption, Elliptic Curve Digital Signature using Mathematica. S. Maria Celestin and K. Muneeswaran implemented text cryptography using ECC by first transforming the message in ASCII values form and mapping into affine points of Elliptic curve by performing point addition of the ASCII value times theGenerator. Sarvana, Suneetha and Chandrasekhar design a method to communicate with multiple parties securely,non-repudiatively in authentic way using ECC with some extra parameters. Jarvinen. K, Helsinki and Skytta. J discusses parallelization of elliptic curve cryptography where the latency of point multiplication is reduced using parallel multiple field multiplier technique with Koblitz curves. Amara M. and Siad A explains the network security role by ECC with smaller key size and compares ECC with RSA and conclude ECC as better choice for encryption. Gopinath Ganapathy and K. Mani design an ECC system using Fuzzy Modular Arithmetic in AT89C51 microcontroller. It observed that, encryption and decryption with fuzzy modular arithmetic consumes less time compared to non-fuzzy modular arithmetic. Scott A. Vansfone provides an overview of current ECC standards and its application and advantages. W. Stalling has given a detailed explanation of various cryptographic techniquein his book Cryptography and Network Security. Loai Tawalbeh, Moad Mowafi and Walid Aljoby used ECCfor performing encryption along with multimedia compression and analyzed encryption efficiency, compression efficiency, codec compliance and security. Balamurugan. R, Kamalakannan. V, Rahul Ganth. D and Tamilselvan. S propose a fast mapping technique using a non singular matrix. First they map the message to points on elliptic curve and later uses ElGamal encryption method to encode the points using a non singular matrix. During decryption inverse of the non singular matrix is used. Megha Kolhekar and Anita Jadhav describe a brief background of encryption/decryption and key exchange using ECC. They have used C++ to implement text encryption. They have used mapping table to map the ASCII value to Elliptic curve coordinate. Reverse mapping is used while decryption.

3. MOTIVATION

Various authors who have implemented text encryption and decryption using ECC have used agreed upon table which consist of characters and ECC coordinates mapping or used the ASCII value of the characters to produce affine elliptic curve coordinates by performing point multiplication operation with generator ‘*G*’ and the corresponding ASCII value of the character. We have come up with a novel idea where use of mapping on common look up table between the sender and receiver has been completely removed. The proposed method is also well suited for large size data as we have designed to encrypt in terms of blocks consisting of multiple characters. Moreover, the algorithm does not limit itself just for English script but it can be used for any script with defined ASCII values

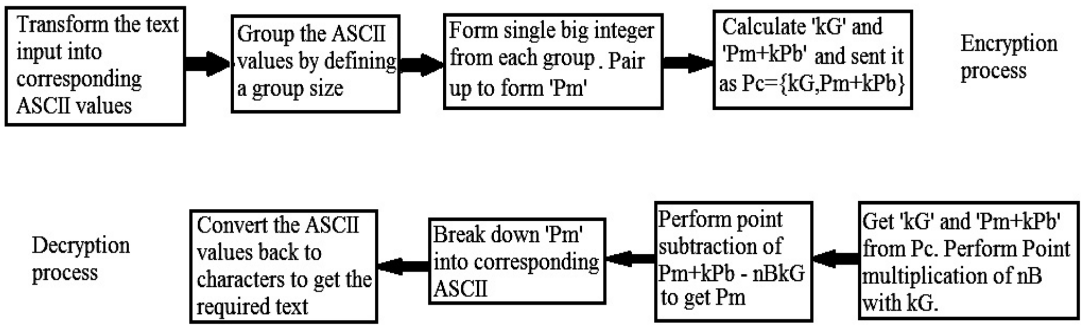

Transform the text input into corresponding ASCII values
Group the ASCII values by defining a group size
Form single big integer from each group. Pair up to form 'Pm'
Calculate 'kG' and 'Pm+kPb' and sent it as Pc={kG,Pm+kPb}
Encryption process
Get 'kG' and 'Pm+kPb' from Pc. Perform Point multiplication of nB with kG.
Perform point subtraction of Pm+kPb - nBkG to get Pm
Break down 'Pm' into corresponding ASCII
Convert the ASCII values back to characters to get the required text
Decryption process

Fig. 2. (a) Block diagram for encryption/decryption process.

4. PROPOSED ALGORITHM

The communicating parties agrees upon an Elliptic curve equation

$$Y^2 = x^3 + ax + b \bmod p$$

with the generator '*G*' and makes the public keys '*Pa*' and '*Pb*' known to all and private keys '*nA*' and '*nB*' are kept secret. Here, we do not map the ASCII values of the characters to affine points of the elliptic curve.We group the ASCII values of the characters and perform cryptographic operation on the group. The size of each group is given by

$$\text{group size} = \text{Length[IntegerDigits[}\ p, 65536]] - 1$$

IntegerDigit [*n*, *b*] function in Mathematica gives a list of the base *b* digits in the integer *n*. Here, we choose base as 65536 because ASCII value is defined till 65535. Length is used to count the number of elements in the given expression. The group size help us to find the maximum number of characters that can be grouped up. Each group is converted into big integer values. We pair up the big integer value and use it as '*Pm*' in the ECC operation. Pairing reduces the operation of mapping to elliptic coordinates and the need to share a common look up table. The whole encryption and decryption is shown as a block diagram in Fig. 2(a).

4.1 Encryption

- Obtain the text to be send.
- Convert to its corresponding ASCII values.
- Partition the ASCII value as

Partition[ASCII values, group size, groupsize, 1, {}] (21)

This operation group the ASCII values with size given by group size with no overlapping and the later sub lists

that have size lesser than group size are left as it is without padding.

- Each group obtained from the above step is converted into big integer values taking base as 65536.

FromDigits[Group of ASCII values, 65536] (22)

- Pad with 32 to the end of the list from the above step if the count of the above list is odd, to make it even for performing complete pairing. Each single pair will be an input to the ECC system as ‘Pm’. We pad with 32 because 32 represents blank space in ASCII code.
- Select random k value, k = Random value with range 1 to n−1. Compute kG and kPb using Point multiplication operation.
- Compute Pm + kPb using point addition or point doubling as required.
- Send Pc = {kG, Pm + kPb} as cipher text to the receiver side.

4.2 Decryption

- Get the cipher text *Pc*.
- Get the left part *kG* and right part *Pm* + *kPb* of the *Pc* separately.
- Multiply with *nB* to the left part and subtract it from the right part to get *Pm*.

  $\{Pm + kPb\} - nBkG = Pm$

  since

  $Pb = nBG.$

Subtraction operation can be converted to addition by multiplying with −1 to the *y* coordinate. This operation can be justified with point addition operation. In point addition we used to get the mirror image point over the *x*-axis. Example:- {97, 24} = {97,−24}.

- The above operation will yield the big integer value which is formed by combining group of ASCII values.

Convert it back to list of ASCII values.

IntegerDigits[big integer, 65536]

IntegerDigits [*n*, *b*] in Mathematica provides a list of the base b digits in the integer *n*. IntegerDigits and

FromDigits function are inverse of each other, so the ASCII values are preserved during encryption and decryption.

- Convert the list of ASCII values to its corresponding characters.

5. SIMULATION OF TEXT ENCRYPTION AND DECRYPTION USING ECC

The simulation was performed using Mathematica version 10 on Lenovo ideapad Z510 laptop with system configuration of i7 processor @ 2.20GHz and 8 GB Ram using 192 bit key length NIST (National Institute of Standards and Technology) recommended Elliptic curve parameter16. The parameters of the simulation are as follows;

a = -3;

b = 2455155546008943817740293915197451784769108058161191238065;

p = 6277101735386680763835789423207666416083908700390324961279;

nB = 2818646689284967968603885680739626753757717668743685336 9;

G = {602046282375688656758213480587526111916698766368846848 18,
174050332293622031404857552280 21

9410364023488927386650641};

Pb = {28030007865416173313773848974350954991247488818907274956 42,
426971802110594428720192929 81

6825304095838300915746390073 9}

5.1 Encryption process

• Input text. The text is shown in Fig. 3(a).

• Its equivalent ASCII values are: {78, 97, 116, 105, 111, 110, 97, 108, 32, 73, 110, 115, 116, 105, 116, 117, 116,

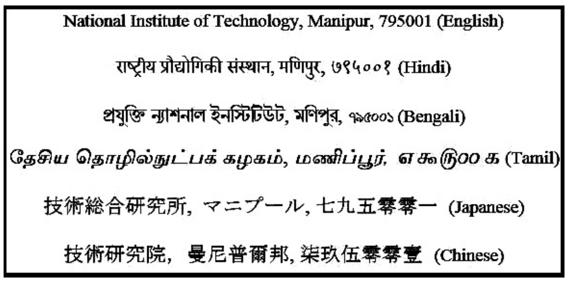
101, 32, 111, 102, 32, 84, 101, 99, 104, 110, 111, 108, 111, 103, 121, 44, 32, 77, 97, 110, 105, 112, 117, 114, 44, 32, 55, 57, 53, 48, 48, 49, 32, 40, 69, 110, 103, 108, 105, 115, 104, 41, 10, 10, 2352, 2366, 2359, 2381, 2335,

2381, 2352, 2368, 2351,. . . , 40, 74, 97, 112, 97, 110, 101, 115, 101, 41, 10, 10, 25216, 34899, 30740, 31350,

38498, 65292, 26364, 23612, 26222, 29246, 37030, 44, 32, 26578, 29590, 20237, 38646, 38646, 22777, 32, 32,

40, 67, 104, 105, 110, 101, 115, 101, 41, 10}

a



National Institute of Technology, Manipur, 795001 (English)

राष्ट्रीय प्रौद्योगिकी संस्थान, मणिपुर, ७९५००१ (Hindi)

প্রযুক্তি ন্যাশনাল ইনস্টিটিউট, মণিপুর, ৭৯৫০০১ (Bengali)

தேசிய தொழில்நுட்பக் கழகம், மணிப்பூர், ௭௯௫௦௦௧ (Tamil)

技術総合研究所, マニプール, 七九五零零一 (Japanese)

技術研究院, 曼尼普爾邦, 柒玖伍零零壹 (Chinese)

Fig. 3. (a) Plain text in different script.

• Group the ASCII values with size calculated as Length [IntegerDigits [*p*, 65536]] - 1 which we get as 11. {{78,

97, 116, 105, 111, 110, 97, 108, 32, 73, 110}, {115, 116, 105, 116, 117, 116, 101, 32, 111, 102, 32}, {84, 101, 99,

104, 110, 111, 108, 111, 103, 121, 44}, . . . , {25216, 34899, 30740, 31350, 38498, 65292, 26364, 23612, 26222,

29246, 37030}, {44, 32, 26578, 29590, 20237, 38646, 38646, 22777, 32, 32, 40}, {67, 104, 105, 110, 101, 115,

101, 41, 10}}

• Convert each group into big integers using From Digits function with base 65536. {11399929092356798485312

5612857907836245105850253422, 1680752752152271159881121378607785507428263635190O8,. . . , 643067

947105844710730362349228912207882422992241O4, 22799458590895254721820532447578822344714}

• Pad with 32 at the end of the above list if the number of term is odd, so that pairing can be done. Pair

them up as ‘*Pm*’, which is one of the parameter used in ECC operation. {11399929092356798485312561

2857907836245105850253422, 168075275215227115988112137860778550742826363519008,. . . , 64306794

710584471073036234922891220788242299224104, 22799458590895254721820532447578822344714, 32}

• After calculating cipher text, *Pc* = {*kG, Pm* + *kPb*} is obtained as

kG = {95058406573787743380879387493754072690640209963862157133, 543754780728205194761539255

6992837333921930872121480709807}

Pm+*kPb* = {{5357129649847875387947498550298509562929834704857479081282, 7750014998021636504

58076998673808830204345207458648302309}, {6179418438352156963426038838668574778107168582785

759775636, 5950440184023478909084289343254612149604486787772222099923}, . . . , {1530967229511514

820723894791502647256505753063012468409818, 2661143118907340934681694726726146262505092101

998749657587}, {4467437980355690213150750580405244103471477917746833402514, 594284377872156

9318800560891052329721902975606688069237688}}

• Send the cipher text *Pc* to the communicating party.

Every run of the program will provide different cipher text even with same input text due the random *k* present in the mathematical operation. We have used only one *kG* value to minimize the size of the cipher text.

5.2 Decryption process

• Obtain the cipher text *Pc* i.e *kG* and *Pm* + *kPb*.

kG = {95058406573787743380879387493754072690640209963862157133, 543754780728205194761539255

6992837333921930872121480709807}

Pm+kPb = {{5357129649847875387947498550298509562929834704857479081282, 7750014998021636504

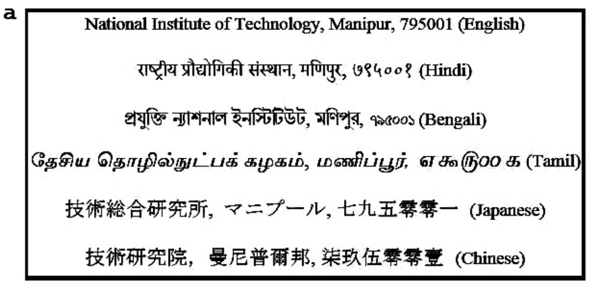
58076998673808830204345207458648302309}, {6179418438352156963426038838668574778107168582785

759775636, 5950440184023478909084289343254612149604486787772222099923},. . . ,{1530967229511514

820723894791502647256505753063012468409818, 2661143118907340934681694726726146262505092101

998749657587}, {4467437980355690213150750580405244103471477917746833402514, 594284377872156

9318800560891052329721902975606688069237688}}


a

National Institute of Technology, Manipur, 795001 (English)

राष्ट्रीय प्रौद्योगिकी संस्थान, मणिपुर, ७९५००१ (Hindi)

প্রযুক্তি ন্যাশনাল ইনস্টিটিউট, মণিপুর, ৭৯৫০০১ (Bengali)

தேசிய தொழில்நுட்பக் கழகம், மணிப்பூர், ௭௯௫00௧ (Tamil)

技術総合研究所, マニプール, 七九五零零一 (Japanese)

技術研究院, 曼尼普爾邦, 柒玖伍零零壹 (Chinese)

Fig. 4. (a) Decrypted text.

• Perform Point multiplication using the private key of the receiver *nB* to *kG* as well as convert the subtraction operation to addition format.

nBkG = {3141192528502843791482798499504492303369782687173663895377, −254483493812166789049

3126265872103594828330153127462384491}

• Perform point addition operation with the above result with the *Pm+kPb*. We get {{11399929092356798485312

5612857907836245105850253422, 168075275215227115988112137860778550742826363519008}, {122768

389944749391054808248629988098406227392397356, 4676976958497714099280437515006237925905355

7678135},. . . ,{36854003571103174246347501364661178996428051260018854, 64306794710584471073036

234922891220788242299224104}, {22799458590895254721820532447578822344714, 32}}

• Convert the above result to ASCII values using Integer Digits function with base 65536.{78, 97, 116, 105, 111,

110, 97, 108, 32, 73, 110, 115, 116, 105, 116, 117, 116, 101, 32, 111, 102, 32, 84, 101, 99, 104, 110, 111, 108,

111, 103, 121, 44, 32, 77, 97, 110, 105, 112, 117, 114, 44, 32, 55,. . . ,10, 25216, 34899, 30740, 31350, 38498,

65292, 26364, 23612, 26222, 29246, 37030, 44, 32, 26578, 29590, 20237, 38646, 38646, 22777, 32, 32, 40, 67,

104, 105, 110, 101, 115, 101, 41, 10, 32}

• Perform conversion operation of ASCII values to characters to get the required message as shown in Fig. 4(a).

5.3 Performance comparison

Table 2. Performance comparison.

Method	Words count	Encryption Time	Decryption time	Cipher Data Size	Mapping	Common look up table
Reference 18	1 word	0.20 seconds	0.30 seconds	1.146 KB	Required	
Reference 7	409 words	1.95 seconds	0.83 seconds	459.118 KB	Required	
Our method	409 words	0.093 seconds	0.14 seconds	21.017 KB	Not Required	Not required

6. SECURITY ANALYSIS

In this section we discuss some analysis of the implemented technique.

6.1 Key space

Security of a cryptographic technique depends a lot on the size of the key used. The algorithm will be known to all. It is always a good choice to have a big key size but we should also keep in mind the computational load when we increase the key size. ECC provides a computationally hard problem called Elliptic Curve Discrete Logarithmic Problem (ECDLP) which help in using a lesser key size compared to other cryptographic technique and still holds the security level high due to ECDLP. In our implementation we have used a 192 bit key length, which is quit good to protect against naive attack. For better security we could increase the key length used for encryption and decryption.

6.2 Key sensitivity

A slight change in the original key should produce a totally different recovered message. Suppose Alice sends a message "Key Sensitivity" to Bob. The original key is nB = 28186466892849679686038856807396267537577176687436853369. The recovered message when the key changes to $nB-1$ = 28186466892849679686038856807396267537577176687436853368 is shown in Fig. 5(a).

6.3 Ciphertext only attack:

Given that the cryptanalyst knows the encryption algorithm and the ciphertext. Until and unless, the cryptanalyst does not have the private key of the receiver the attacker can't reveal the plain text. Applying Brute Force attack would not be of much help when that key size is very large as it will take lots of time in term of years. So, even if the analyst is able to decrypt it, the value of the information will be no more at that time.

6.4 Known plaintext attack:

Given that the cryptanalyst knows the encryption algorithm, ciphertext and one or more plaintext-ciphertext pairs formed with the secret key. Since, the implementation generate a different cipher text for the same message due to the random k used in the operation. Known Plaintext attack cannot do any harm.

6.5 Time complexity:

The best known attack on ECC is Pollard's Rho method and Pollard Lambda method. Pollard's Rho method is expected to find the private key at most a constant time Sqrt[*N*] steps, where *N* is the cyclic order of the Elliptic curve with G as Generator. Pollard's Lambda method is similar to Pollard's Rho method but it uses many starting points to find a match. Pollard's lambda method also expects to find the private key at most a constant time Sqrt[*N*] steps.

If implemented in parallel, the running time to find the private key can be reduced. Both these methods are probabilistic method i.e. they got a high probability but does not guarantees to finish in a constant time of Sqrt[*N*] steps. For a 192 bit Elliptic Curve that we have used,

N value is 6277101735386680763835789423176059013767194773182842284 081.

$\sqrt{N}$steps = 7.92282* 10^{28} steps

Assume each step takes just 0.0000001 second, still it will take around

1.90148 * 10^{23} days

to find the private key. By this time, the value of the message will be no more and the communicating party may have used a new cryptographic technique or may have changed their keys.

7. CONCLUSION

In this given paper we have implemented a new technique to perform text cryptography using ECC. Here, we divide the text ASCII values into groups, where group size is determined using ' *p*' value of ECC parameters with a base which is greater than the maximum ASCII value present in the script. Big integers are formed using each group and the group were paired and fed as '*Pm*' into ECC operation. This process helps in removing the costly operation of mapping the characters to coordinates of Elliptic curve as well as the need to share the common look up table. The proposed algorithm can be used for any script with defined ASCII value. From the performance comparison table we can say that our proposed algorithm has got lot of positive aspect. Encryption and decryption operation is performed very swiftly even with large number of words as input, provides smaller size cipher text compared to other technique which greatly helps in saving bandwidth while sending and we don't require mapping and common look up table. ECC provide a better security with lesser key size compared to the very successful RSA. Elliptic curve discrete logarithm problem is very hard to solve, this property is used in ECC. As ECC provides equal security like other cryptographic system but with less key size, it is very suitable for devices which have power, storage and processing limitation.

REFERENCES

[1] Victor S. Miller, Use of Elliptic Curves in Cryptography, *Advances in Cryptology-CRYPTO'85 Proceedings*, Springer, vol. 218, pp. 417–426, December (2000).

[2] Neal Koblitz, Elliptic Curve Cryptosystems, *Mathematics of Computation*, vol. 48, issue 177, pp. 203–209, January (1987).

[3] Neal Koblitz, Alfred Menezes and Scott Vanstone, The State of Elliptic Curve Cryptography, Designs, Codes and Cryptography, vol. 19, issue 2–3, pp. 173–193, (2000).

[4] Darrel Hankerson, Alfred Menezes and Scott Vanstone, Guide to Elliptic Curve Cryptography, Springer (2004).

[5] Lawrence C. Washington, Elliptic Curves Number Theory and Cryptography, Taylor & Francis Group, Second Edition (2008).

[6] Jorko Teeriaho, Cyclic Group Cryptography with Elliptic Curves, Brasov, May (2011).

[7] S.Maria Celestin Vigila and K. Muneeswaran, Implementation of Text based Cryptosystem using Elliptic Curve Cryptography, *International Conference on Advanced Computing, IEEE*, pp. 82–85, December (2009).

[8] D. Sravana Kumar, Ch. Suneetha and A. Chandrasekhar, Encryption of Data Using Elliptic Curve Over Finite Fields, *International Journal of Distributed and Parallel Systems (IJDPS)*, vol. 3, no. 1, January (2012).

[9] K. Jarvinen, Helsinki and J. Skytta, On Parallelization of High-Speed Processors for Elliptic Curve Cryptography, VLSI Systems, *IEEE Transaction*, vol. 16, issue 9, pp. 1162–1175, August (2008).

[10] M. Amara and A. Siad, Elliptic Curve Cryptography and its Applications, *7th International Workshop on Systems, Signal Processing and their Applications*, pp. 247–250, May (2011).

[11] Gopinath Ganapathy and K. Mani, Maximization of Speed in Elliptic Curve Cryptography Using Fuzzy Modular Arithmetic over a Micro-controller based Environment, *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, (2009).

[12] Scott A. Vansfone, Elliptic Curve Cryptography-The Answer to Strong, Fast Public-Key Cryptography for Securing Constrained Environments, *Information Security Technical Report*, vol. 2, no. 2, pp. 78–87, (1997).

[13] O. Srinivasa Rao and S. Pallam Setty, Efficient Mapping Methods for Elliptic Curve Cryptography, *International Jounal of Engineering Science and Technology*, vol. 2(8), pp. 3651–3656, (2010).

[14] Williams Stallings, Cryptography and Network Security, Prentice Hall, 4th Edition, (2000).

[15] Lo'ai Tawalbeh, Moad Mowafi and Walid Aljoby, Use of Elliptic Curve Cryptography for Multimedia Encryption, *IET Information Security*, vol. 7, issue 2, pp. 67–74, (2012).

[16] www.csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf, (1999).

[17] R. Balamurugan, V. Kamalakannan, D. Rahul Ganth and S. Tamilselvan, Enhancing Security in Text Messages Using Matrix based Mapping and ElGamal Method in Elliptic Curve Cryptography, *International Conference on Contemporary Computing and Informatics*, *IEEE*, pp. 103–106, November (2014).

[18] Megha Kolhekar and Anita Jadhav Implementation of Elliptic Curve Cryptography on Text and Image, *International Journal of Enterprise Computing and Business Systems*, vol. 1, issue 2, July (2011).