Design of a Modified Encryption Algorithm for Data Transmission over the Network

[1]Deepa Soni, [2]Amrendra Kumar Singh
[1]Student (M. tech), [2]Assistance Professor (Dept. of Information Technology)
[1]E-Security (Information Technology)
[1,2]Bhilai Institute of Technology, Durg, India

Abstract: In today's digital world every data is on air even a person's identification has been done by their digital signature. Hence, for everyone it is a major concern to protect their data and identification with the use of limited resources on digital world. The encrypted message is confidential but when it comes to non-repudiation only encryption is not sufficient to provide this service so we need Digital Signature. If we want to use the digital signature or want to perform encryption/decryption then it is must to frequently change the Key value, in RSA key generation is costly so feel need of the algorithm in which key generation is less costly so we can change key frequently.

Keywords: RSA, Digital Signature, ECC, Encryption, Decryption.

1. Literature Review

In order to gain more clear understanding of the elliptic curve algorithm and RSA algorithm which is used for digital signatures and key exchange protocols, several attributes are studied. These attributes include parameters, key generation, encryption, signatures, authentication and verifications. For this purpose, ANSI X9.62 [1] and IEEE 1363 [2] are critical standards which provide the necessary information about the algorithms and protocols.

1.1 Conventional ECC Digital Signature Algorithm

Don Johnson *et al*, (2000) [3] described the Elliptic Curve Digital Signature Algorithm, ECDSA stands for Elliptic Curve Digital Signature Algorithm. It is the implementation of digital signature with the help of elliptic curves. It is an accepted standard by ANSI, NIST, ISO and IEEE. This paper discusses the various properties of elliptic curves, the concept behind the operations carried out on elliptic curves, the algorithm proposed for digital signature over the elliptic curves and also the discrete logarithmic problem associated with it.

Kristin Lauter *et al*, (2004) [4] proposed the advantages of Elliptic Curve Cryptography like ECC uses smaller keys, compact size certificates and quick verification of certificates makes overall encryption system less costly as compare to other public key encryption algorithms. The superiority of ECC is appropriate for the environment where processing power, storage power, band width or power consumption is limited. So ECC has all that features that any encryption system should possess to secure the communication and message verification and hence it is becoming more and more popular encryption system among wireless industry. Wireless industry such as Motorola, Docomo, and RIM. Major computer companies such as IBM, Sun Microsystems, Microsoft, and Hewlett-Packard are all investing in ECC.

1.2 Modified RSA Algorithm

M. Preetha *et al*, (2013) [5] proposed performance analysis of RSA algorithm public key algorithm RSA and enhanced RSA are compared analysis is made on time based on execution time. The idea of his work is to enhance the security of RSA algorithm and has done it by implementing and comparing the RSA and Optimal Asymmetric Encryption Padding (OAEP), OAEP is a method for encoding message. The technique of encoding a message with OAEP and then encrypting it with RSA is provably secure in the random oracle model. Informally, this means that if hash functions are truly random, then an adversary who can recover such a message must be able to break RSA.

Fausto Meneses *et al*, (2016) [6] proposed research that is focused on optimizing the RSA encryption algorithm. To achieve this, they designed and implemented a generic solution capable of encrypt and decrypt information, increasing efficiency and security of messages transmitted over the network. This solution included the review of the model, optimization of the mathematical expression model, which has improved the encryption method, and implementation of algorithms for secure transmission of messages on the network. They used Netbeans 8.0 which is free software that allowed the development of a new API with Java. Within this API we declared the method of the new RSA algorithm. The results show the functionality, security and usability of our study, but especially show quantitatively that the algorithm has been optimized.

Nikita Somani *et al*, (2014) [7] Proposed Improved RSA Cryptographic System, the RSA cryptosystem and its variants. The proposed algorithm has speed improvement on the decryption side of RSA algorithm by using the concept of Chinese remainder theorem and the method also improves the security of RSA algorithm by avoiding some attacks that are possible on RSA algorithm like common modulus attack, chosen ciphertext attack, timing attack and known plaintext attack.

1.3 Comparison of ECC and RSA Algorithm

Nicholas Jansma *et al*, (2004) [8] proves that in his comparison that the RSA is slower its key is greater in size as compared to ECC. He has compared key generation, signature generation and signature verification component of both RSA and ECC independently and suggested that RSA algorithm is much better choice for the application environment where message verification is more frequent need than signature generation. According to him, for digital signature creation in terms of time RSA is comparable to ECC and for signature verification RSA is faster than ECC as his result shows.

Dindayal Mahto *et al*, (2017) [9], this paper presented a comparative analysis of RSA and ECC. The experimentation was conducted for finding time lapse during encryption, decryption by RSA and ECC on three sample input data of 8 bits, 64 bits, 256 bits with random keys based on NIST recommendation. Based on experimentation, it was found that ECC outperforms RSA regarding operational efficiency and security with lesser parameters. An ECC is particularly most suitable for resource constraint device

Kamlesh Gupta *et al*, (2011) [10], performed comparative study of ECC and RSA to analyse their behaviour, also provide us information where the use of ECC is suitable like in portable devices and for wireless devices. An extensive review of most important ECC implementation in java is presented.

1.4 Modified ECC Algorithms

Katsuyuki Okeya *et al*, (2001) [11] solve the problem that Lim and Hwang stated: "Montgomery's method is not a general algorithm for elliptic scalar multiplication in GF(pn), since it can't compute the y-coordinate of kP." Author has proved that Montgomery's method is indeed a general algorithm. He has compared his proposed algorithms with the traditional scalar multiplication algorithm and his analysis shows that scalar multiplication on a Montgomery-form elliptic curve require no precomputation and is faster than that of the window method.

Ankita Jena *et al.* (2013) [12] proposed an improved Elliptic Curve Cryptography and her proposed algorithm is more efficient in term of execution time as compared to original ECDSA. She implemented the ECDSA in Modified Jacobian co-ordinates with point multiplication done using Montgomery Scalar Multiplication it is efficient than the one proposed by Don Johnson and Alf. Her algorithm takes 4 second to complete the execution while the original ECDSA takes 9 second to complete the execution she has improved almost 55.55% efficiency of the original ECDSA.
Al Imem Ali *et al*. (2015) [13] proposed a modified Montgomery method for reducing the execution time and signature generation time. He has used Open-SSL for comparing the performance of the RSA and ECDSA Digital signature in his comparison he found that RSA key generation take more time, execution time of RSA is more and verification of the RSA is less. He has integrated Montgomery method to reduce the number of operations during the computation task and succeeded to accelerate the ECDSA sign and/or verification process.

Jiahong *et al.* (2009) [14], describes an efficient hardware implementation of Elliptic Curve Cryptography (ECC) over *GF*(*p*) in the article named "A Fast Hardware Implementation of Elliptic Curve Cryptography". The fast mechanism for accelerating the elliptic curve point operation formulae by using modified Jacobian coordinates. The mechanism reduces the number of multiplication in double procedure (squaring) and since the double procedure appears often in point multiplication, a performance gain is achieved. The final result on Virtex II XC2V250 indicates the computing time of multiplication, power consumption and slices are 1.5 × 10, 50mW and 92% respectively. By using the Jacobian coordinates, the power acceleration of the operation is increased. The main advantage is Jacobian coordinates, which deals with slices, power consumption and countermeasure design modification.

Meloni *et al.* (2006) [15], proposed a new improved and secure point multiplication algorithm in the paper named "Fast and Secure Elliptic Curve Scalar Multiplication over Prime Fields Using Special Addition Chains". This research has been made on a particular kind of addition chains consisting only addition and not using point doubling, this algorithm provides a natural protection against side channel attacks. Moreover, they propose new addition formulae that take into account the specific structure of those chains making point multiplication very fast.

2. Proposed Algorithm

We have considered that time is an important factor in the field of encryption decryption process so we ponder to make an algorithm that take less time for digital signature while maintains the same or higher level of security. We have studied two algorithms for encryption, decryption, key generation, signature generation and signature verification RSA and ECC as earlier described and introducing a new algorithm that is the combination of the ECC and RSA and we will call it as Modified Elliptic Curve Cryptography (MECC).

2.1 MECC Key Generation Algorithm

We have used both ECC and RSA algorithm for MECC key generation. Firstly we have generated public/private key by using ECC algorithm then have generated public/private key by using RSA algorithm.

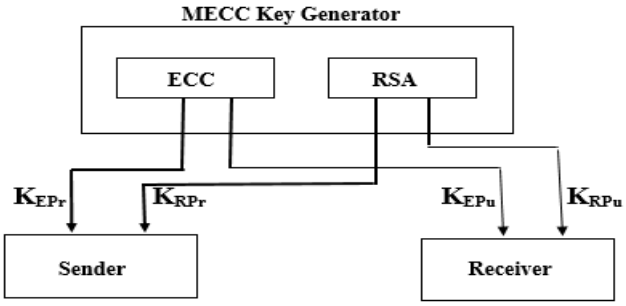

Figure 2.1 MECC Key Generation

- **MECC Public Key Generation**

For public key generation, we have appended public key which is generated by ECC with public key generated by RSA. Hence we get the public key of MECC.

1) Apply the ECC algorithm & generate the public key value (K_{EPu}).
2) Apply the RSA algorithm & generate the public key value (K_{RPu}).
3) Append the E_{PU} to R_{PU} and get the public key value for K_{MPu}.

$$K_{MPu.} = K_{EPu} \parallel K_{RPu}$$

4) Repeat the step 1 to step3 to get new instance of MECC's private key generation
5) Stop.

- **MECC Private Key Generation**

For private key generation, we have appended private key which is generated by ECC with private key generated by RSA. Hence we get the private key of MECC.

1) Apply the ECC algorithm & generate the private key value (K_{EPr}).
2) Apply the RSA algorithm & generate the private key value (K_{RPr}).
3) Append the K_{EPr} to K_{RPr} and get the public key value for K_{MPr}.

$$K_{MPr} = K_{EPr} \parallel K_{RPr}$$

4) Repeat the step 1 to step 3 to get new instance of MECC's private key generation.
5) Stop.

2.2 MECC Encryption and Decryption Algorithm

We have used both ECC and RSA algorithm for MECC encryption/decryption. Firstly we perform encryption/decryption of the message by using ECC algorithm then perform encryption/decryption by using RSA algorithm.

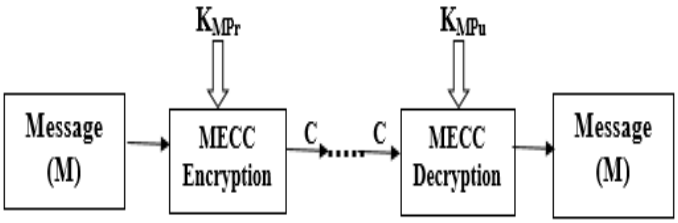

Figure 2.2 MECC Encryption and Decryption Process

- **MECC Encryption**

For encryption, we encrypt message by ECC with ECC's private key then we encrypt the encrypted message by RSA with RSA's private key. Hence we get the encrypted message by MECC.

1) Apply the ECC encryption algorithm on message (M) using private key K_{EPr}, let the output is C_E.

$$C_E = E(K_{EPr}, M)$$

2) Apply the RSA encryption algorithm on C_E using private key K_{RPr} ,let output is C_{ER}.

$$C_{ER} = E(K_{RPr}, C_E)$$

3) Repeat the step 1 & 2 to get the message encrypted by MECC.
4) Stop.

- **MECC Decryption**

For decryption, we firstly decrypt the message by RSA with RSA's public key then we decrypt the decrypted message by ECC with ECC's public key. Hence we get the decrypted message by MECC.

1) Apply the RSA decryption algorithm on encrypted text C_{ER} using private key K_{RPu}.

$$C_E = D(K_{RPu}, C_{ER})$$

2) Apply the ECC decryption algorithm on C_E using public key K_{EPu}.

$$M = D(K_{EPu}, C_E)$$

3) Repeat the step 1 & 2 to get the message decrypted by MECC.
4) Stop.

2.3 MECC Signature Generation

We have used both ECC and RSA algorithm for MECC Digital Signature generation. Firstly we generate digital signature by using ECC digital signature algorithm then generate digital signature by using RSA digital signature algorithm.

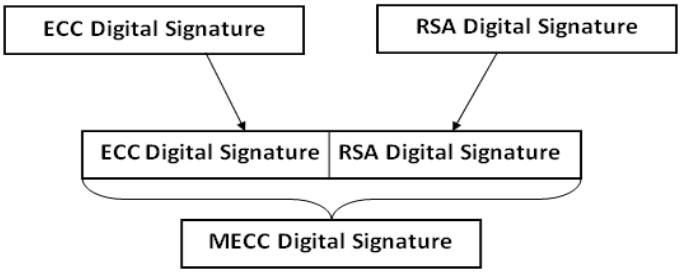

Figure 2.3 MECC Digital Signature Generation

1) Apply the ECC Digital signature algorithm & generate the signature let the generated signature is DS_E.
2) Apply the RSA Digital signature algorithm & generate the signature let the generated signature is DS_R.
3) Append the DS_E to DS_R and get the Digital signature value for ME_{DS}

$$DS_M = DS_E \parallel DS_R$$

4) Repeat the step 1 to step3 to get new instance of MECC Digital Signature (DS_M).
5) Stop.

2.4 MECC Signature Verification

As we have used both ECC and RSA algorithm for MECC Digital Signature generation so we have to use both algorithm for verification of digital signature also.

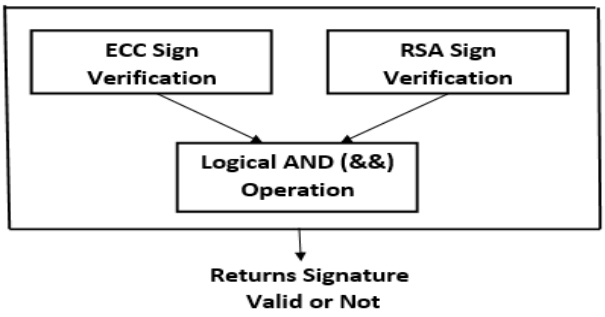

Figure 2.4 MECC Digital Signature Verification

1) Verify the ECC Digital signature DS_E by ECC verification algorithm, let output is V_E.
2) Verify the RSA Digital signature DS_R by RSA verification algorithm, let output is V_R.
3) Perform Logical AND (&&) operation on the V_E and V_R.

$$V_M = V_E \ \&\& \ V_R$$

4) Repeat the step 1 to step 3 for verification of digital signature by MECC verification (V_M).
5) Stop.

3. Result

The algorithms are coded in Java and the implementation of these algorithms is performed through NetBeans 8.2 IDE. The implementation is carried step by step by varying the bits used for encryption, decryption, key generation, signature generation. Performance Comparison among the algorithms is shown by the help of the line graphs. Different values are plotted by the help of different colours which are pre-decided and same colour has been used for plotting all the graphs they are as followed:

- **Red Line:** Shows the performance of the RSA algorithm.
- **Green Line:** Shows the performance of the ECC algorithm.
- **Yellow Line:** Shows the performance of the MECC algorithm.

3.1 Graph of RSA, ECC and MECC Key Generation: This figure shows the line graph generated for the purpose of comparison of RSA, ECC and MECC performance for Key Generation task. We can easily analyse that the performance of RSA Key Generation algorithm is become worse as the key size increased.

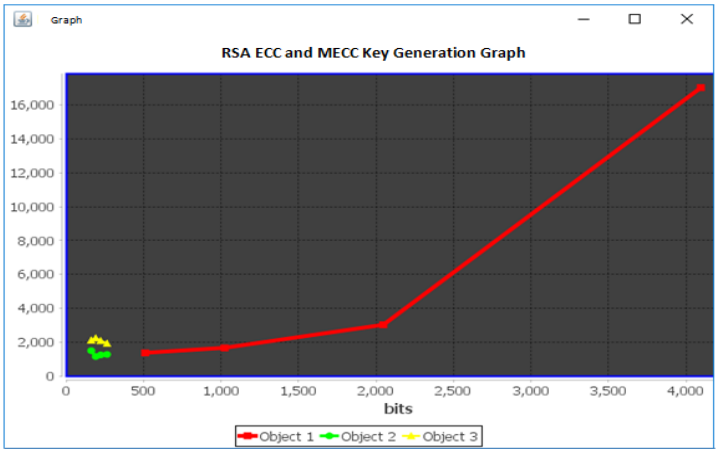

Figure 3.1 Graph of RSA, ECC and MECC Key Generation

3.2 Graph of ECC and MECC Key Generation: Since the performance of the ECC and MECC is extremely good in comparison with RSA and the values of ECC and MECC are not visible clearly in Figure 3.1, so we have took another snap shot of the graph to show the values of ECC and MECC, and from the shown graph we can see that MECC takes slightly more time for key generation in comparison to ECC.

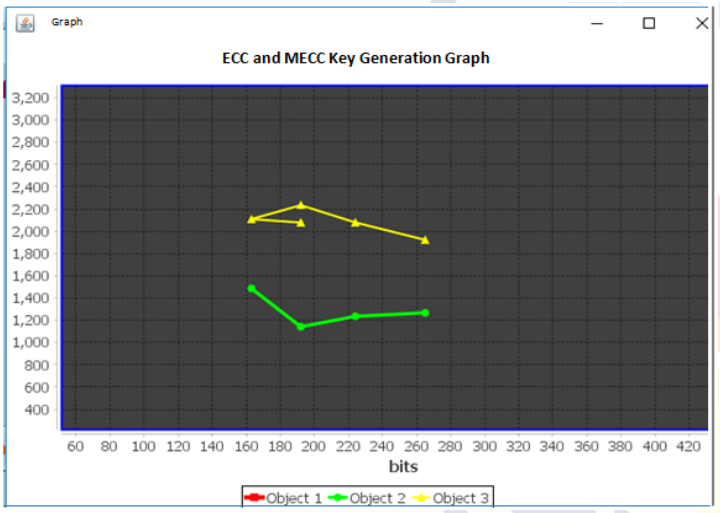

Figure 3.2 Graph of ECC and MECC Key Generation

3.3 Graph of RSA, ECC and MECC Sign Generation: This graph shows the performance for Signature Generation task, of RSA, ECC and MECC and from graph it is clear that the performance of the RSA sign generation algorithm is good as we increase the key size of algorithm.

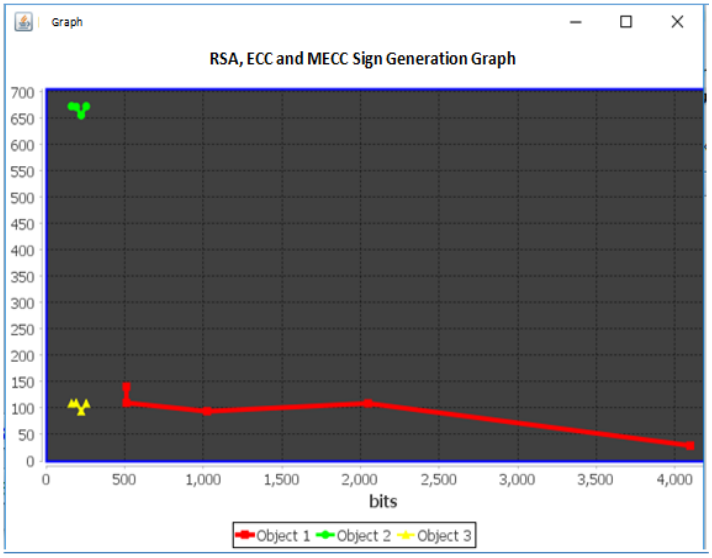

Figure 3.3 Graph of RSA, ECC and MECC Sign Generation

3.4 Graph of ECC and MECC Sign Generation: Since the key size of the ECC and MECC is extremely small in comparison with RSA and the values of ECC and MECC are not shown properly in figure 3.3 so we have zoomed the graph just to show the values of ECC and MECC, and from graph we can see that MECC takes much more time for signature generation in comparison to ECC.

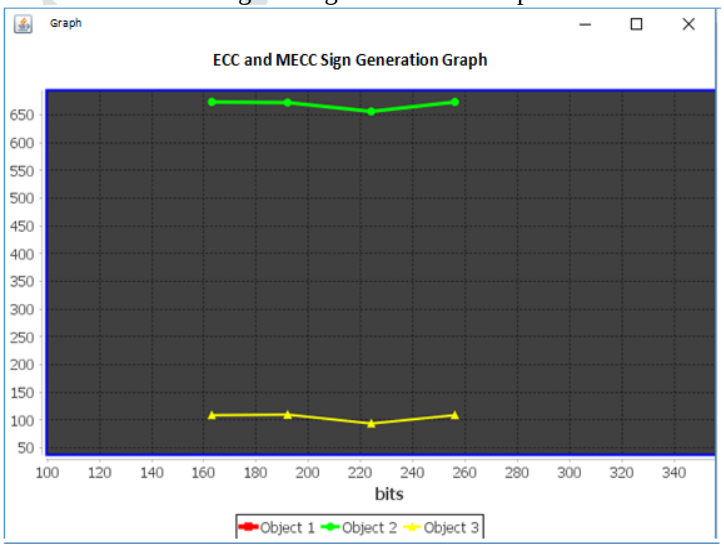

Figure 3.4 Graph of ECC and MECC Sign Generation

3.5 Graph of RSA, ECC and MECC Sign Verification: This figure shows the line graph generated for the purpose of comparison of RSA, ECC and MECC performance for Signature Verification task, and by graph shown we can analyse that the performance of the RSA sign verification algorithm is best as compare to ECC and MECC.

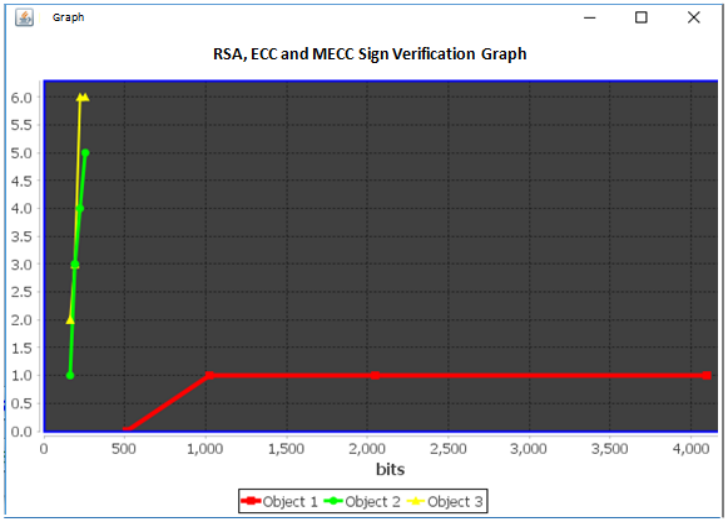

Figure 3.5 Graph of RSA, ECC and MECC Sign Verification

3.6 Graph of ECC and MECC Sign Generation: Since the key size of the ECC and MECC is extremely small in comparison with RSA and the values of ECC and MECC are not shown properly in figure 3.5 so we have zoomed the graph just to show the values of ECC and MECC, and from graph we can see that MECC and ECC takes approximately same time for signature verification.

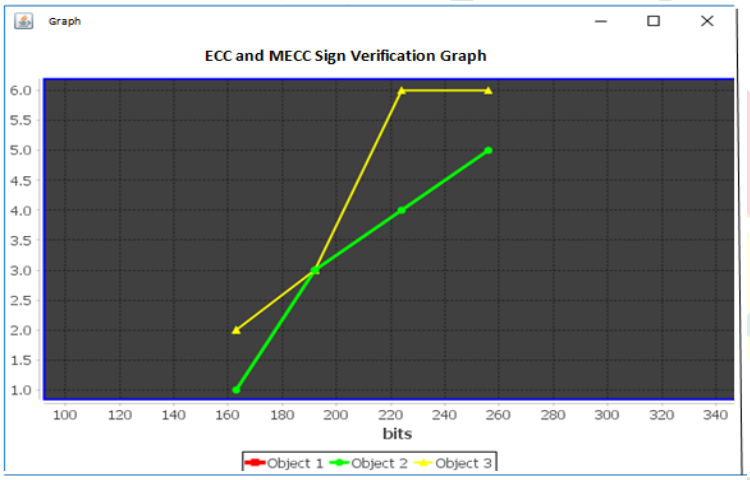

Figure 3.6 Graph of ECC and MECC Sign Verification

4. Conclusion

Proposed algorithm takes less time for execution when we need to generate new key frequently and also the propose algorithm is more secure than the RSA because it is the combination of ECC and RSA algorithm.

5. Reference

[1] ANSIX9.62. Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). New York, USA: American National Standards Institute, 1999.

[2] IEEE-P1363-2000. IEEE Standard Specifications for Public-Key Cryptography. MD, USA: IEEE Computer Society Press, Silver Spring, 2000.

[3] Don Johnson and Alfred Menezes, “The Elliptic Curve Digital Signature Algorithm ECDSA”, 2000, Springer.

[4] Kristin lauter, “The advantages of Elliptic Curve Cryptography for Wireless Security”, IEEE Wireless Communications, February 2004.

[5] M. Preetha, M. Nithya, "A STUDY AND PERFORMANCE ANALYSIS OF RSA ALGORITHM ", IJCSMC, Vol. 2, Issue. 6, June 2013, pg.126 – 139.

[6] Fausto Meneses et.al, "RSA Encryption Algorithm Optimization to Improve Performance and Security Level of Network Messages ",IJCSNS International Journal of Computer Science and Network Security, VOL.16 No.8, August 2016.

[7] Nikita Somani, Dharmendra Mangal, "An Improved RSA Cryptographic System", International Journal of Computer Applications (0975 – 8887) Volume 105 – No. 16, November 2014.

[8] Nicholas Jansma and Brandon Arrendondo “Performance Comparison of Elliptic Curve and RSA Digital Signatures”, April 28, 2004.

[9] Dindayal Mahto,”RSA and ECC: A Comparative Analysis”, International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 19 (2017) pp. 9053-9061.

[10] Kamlesh Gupta, Sanjay Silakari, “ECC over RSA for Asymmetric Encryption: A Review”, IJCSI, Vol. 8, Issue 3, No. 2, May 2011.

[11] Katsuyuki Okeya and Kouichi Sakurai “Efficient Elliptic Curve Cryptosystem from a Scalar Multiplication Algorithm with Recovery of the y-Coordinate on a Montgomery-Form Elliptic Curve” Springer, 2001.

[12] Ankita Jena “Improved Authentication Mechanism Based On Elliptic Curve Cryptography”, May 2013.

[13] Al Imem Ali, “Comparison and Evaluation of Digital Signature Scheme Employed in NDN Network”, International Journal of Embedded systems and Applications (IJESA), Vol.5, No.2, June 2015, DOI: 10.512/ijesa.2015.5202.

[14] Jiahong Zhang, Tinggang Xiong, Xiangyan Fang, “A Fast Hardware Implementation of Elliptic Curve Cryptography” Information Science and Engineering (ICISE), 2009 1st International. Conference ,DOI: 10.1109/ICISE.2009.29.

[15] Nicolas Meloni “Fast and Secure Elliptic Curve Scalar Multiplication Over Prime Fields Using Special Addition Chains”, January 2006.