International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 19 (2017) pp. 9053-9061

RSA and ECC: A Comparative Analysis

Dindayal Mahto
Department of Computer Applications, National Institute of Technology Jamshedpur, Adityapur, Saraikella-Kharsawan, Jharkhand, India.

Orcid Id: 0000-0001-5599-4928

Dilip Kumar Yadav
Department of Computer Applications, National Institute of Technology Jamshedpur, Adityapur, Saraikella-Kharsawan, Jharkhand, India.

Orcid Id: 0000-0002-1334-7500

Abstract

This paper presents a comparative analysis of RSA (Rivest Shamir Adleman) and ECC (Elliptic Curve Cryptography). In the current digital world and public-key cryptography segment, the majority of deployments are fulfilled by the RSA based cryptosystems. Cryptosystems based on elliptic curves emerge as an alternative to the RSA cryptosystems. The security of the RSA cryptosystem is based on the integer factorization problem (IFP) whereas the security of ECC is based on the elliptic curve discrete logarithm problem (ECDLP). The significant attraction towards ECC is that the best-known algorithm for solving the ECDLP takes full exponential time while for solving IFP of RSA takes sub-exponential time. This analysis suggests that ECC takes less memory than RSA and is better than RSA, especially on memory-constrained devices.

Keywords: RSA, Elliptic Curve Cryptography, ECDLP, IFP, Public-Key Cryptography.

INTRODUCTION

Nowadays we live in a digital world where a majority of our messages or information gets exchanged between communicating users or systems immediately through digital devices and digital network. However, the Internet, being an open-ended architecture, has some flaws through which eavesdroppers perform cyber attacks on communicated message. Using cryptographic techniques, we can curb on such type of attacks.

Cryptography is an art and a science of secret writing. It is of two types: symmetric-key/private-key cryptography and asymmetric-key/public-key cryptography. Symmetric-key cryptosystems are encryption/decryption systems which provide message confidentiality only. An asymmetric-key cryptography technique provides confidentiality, integrity, and authentication of traveling/storage message. Although symmetric-key cryptosystems are faster and efficient than asymmetric-key cryptosystems, however, they suffer from key distribution and key management problems, whereas, asymmetric-key cryptosystems provide an excellent way to distribute key and to avoid key distribution and key management problems of symmetric-key cryptosystems.

Concerning security of RSA and ECC, the fastest algorithm (Pollard's rho algorithm) known for solving the ECDLP takes full exponential time, which has an expected running time of $\sqrt{\pi n/2}$. As of 2003, the largest ECDLP instance solved with Pollard's rho algorithm is an elliptic curve over a 109-bit prime field. The best known generic factoring method is Pollard's general number field sieve (NFS). The heuristic expected runtime needed for the NFS to find a factor of the composite number n is L[n] = [1/3, 1.923]. The largest integer factored using the NFS takes sub-exponential time, is RSA200, a 200-digit number (665 bits) which was factored in May 2005 [1]. This means that, for the same level of security, significantly smaller parameters can be used in ECC than RSA. For example, to achieve 112 bits of security level, RSA algorithm needs a key size of 2048 bits, while ECC needs a key size of 224 bits [2] as shown in Table 1 and Figure 1.

A comparative analysis of RSA and ECC is presented on the basis encryption and decryption times for the data of 8 bits, 64 bits, and 256 bits.

Table 1: NIST Recommended Security Bit Level

Security Bit Level	RSA	ECC
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

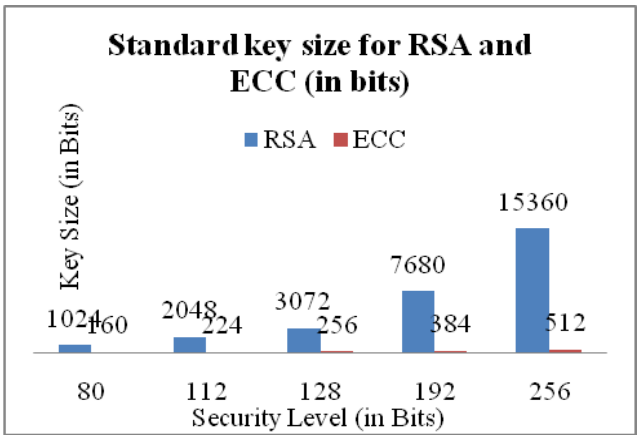

Figure 1: NIST Recommended Security Bit Level

METHODS

In the literature, some of the authors have presented the comparative/security/performance analysis of RSA and ECC with different parameters of measurements. Gura et al. [3] compared point multiplication operation of an elliptic curve over RSA and ECC on two 8-bit processor computer systems and they found that on both systems, ECC-160 point multiplication is more efficient than the RSA-1024 private-key operation. Bos et al. [4] assess the risk of usage of a key on the basis of key length of RSA and ECC, and they conclude that till 2014, use of 1024-bit RSA provides some small risk while 160-bit ECC over a prime field may safely be used for a much more extended period. Kute et al. [5] concluded RSA is faster than ECC, but security wise ECC outperforms RSA. Jansma et al. [6] compare the usages of digital signatures in RSA and ECC and suggest, RSA may be a good choice for the applications, where verification of message is required more than a generation of the signature. Alese et al. [7] suggested that currently, RSA is stronger than ECC although they also indicated ECC outperforms than RSA in future. Mahto et al. [8, 16-21] demonstrate that ECC outperforms regarding operational efficiency and security over RSA.

RSA

RSA [9] is considered as the first real life and practical asymmetric-key cryptosystem. It becomes de facto standard for public-key cryptography. Its security lies with integer factorization problem. RSA's decryption process is not efficient as its encryption process. Many researchers have proposed to improve the efficiency of RSA's decryption using Chinese Remainder Theorem (CRT). Verma et al. [22] proposed a model to improve decryption time of the RSA using CRT. They also proposed to generate large modulus and cryptographic keys with small order of a matrix.

For better and stronger security of data, bigger key sizes require, which means more overhead on the computing systems. Nowadays small devices are playing an important role in the digital world, which has less memory but needs security to cope with market demand. In this scenario, RSA becomes second thoughts.

RSA Algorithm

Key Generation

Step I. Select p, q p and q both are primes, $p \neq q$

Step II. Calculate $n = pq$

Step III. Calculate $\Phi(n) = (p-1)(q-1)$

Step IV. Select integer e $\gcd(\Phi(n), e) = 1; 1 < e < \Phi(n)$

Step V. Calculate d $d \equiv e^{-1} \pmod{\Phi(n)}$

Step VI. Public key PU = $\{e, n\}$

Step VII. Private key PR = $\{d, n\}$

Encryption

Step I. Plaintext: $M<n$

Step II. Ciphertext: $C=M^e \bmod n$

Decryption

Step I. Ciphertext: C

Step II. Plaintext: $M=C^d \bmod n$

Here, key generation is to be done by each party, so that they can communicate each other securely. In the RSA algorithm, 'e' is for encryption, should be chosen such that gcd(Φ(n), e) is equal to 1. Once 'e' is selected, corresponding, 'd' that is for decryption should be generated with the help of finding the inverse of 'e' mod Φ(n).

In encryption process, a sender has to encrypt the message (i.e., in decimal digit) with the help of receiver's public key, i.e., 'e' and 'n'.

In decryption process, the receiver has to decrypt the ciphertext with the help of his private key, i.e., 'd' and 'n'.

ECC

An ECC over a prime field is defined by following general equation in two variables with coefficients.

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

where, $4a^3 + 27b^2 \neq 0$.

ECC is other promising asymmetric key cryptosystems, independently coined by Miller [10] and Koblitz [11] in the late 1980s. This type of systems is most suitable for memory constraint devices such as Palmtop, Smartphone, Smartcards, etc. An ECC requires comparatively less or smaller parameters for encryption and decryption than RSA, but with equivalent levels of security.

ECC Algorithm

Global Public Elements

Step I. $E_q(a, b)$ elliptic curve with parameters a, b, and q, where q is a prime or integer of the form 2^m.

Step II. G point on elliptic curve whose order is large value n

User Alice Key Generation

Step I. Select private key n_A; $n_A < n$

Step II. Calculate public key P_A

Step III. $P_A = n_A G$

User Bob Key Generation

Step I. Select private key n_B; $n_B < n$

Step II. Calculate public key P_B

Step III. $P_B = n_B G$

Calculation of Secret Key by User Alice

Step I. $K = n_A P_B$

Calculation of Secret Key by User Bob

Step I. $K = n_B P_A$

Encryption by Alice using Bob's Public Key

Step I. Alice chooses message P_m and a random positive integer 'k'

Step II. Ciphertext: $C_m = \{ kG, P_m + kP_B \}$

Decryption by Bob using his own Private Key

Step I. Ciphertext: C_m

Step II. Plaintext: $P_m = P_m + kP_B - n_B(kG)$

$= P_m + k(n_B G) - n_B(kG)$

Here, P_m is a (x,y) point encoded with the help of plaintext message 'm'. The P_m is the point used for encryption and decryption.

An Illustration of Elliptic Curve Cryptography

This illustration presents a data communication security model for a message of 64-bits using ECC.

Key Exchange using Elliptic Curve Diffie-Hellman Algorithm [12]

Here, global parameters of ECC are:

Prime number q=8209, a=2, b=7, G=(4, 1313), h=1% of secret key (ie.K(x)), for encoding and decoding of message in elliptic curve. Based on global parameters, the elliptic curve's equation becomes:

$$y^2 \bmod 8209 = (x^3 + 2x + 7) \bmod 8209 \quad (2)$$

Steps for key exchange:

Step I. Private Key of Alice is a random value: d_A=4706

Step II. Public Key of Alice is:

$P_A(x, y) = d_A * G(x, y)$

$= 4706 * (4, 1313)$

$= (7926, 5458)$

Step III. Private Key of Bob is a random value:

$d_B = 4802$

Step IV. Public Key of Bob is:

$P_B(x, y) = d_B * G(x, y)$

$= 4802 * (4, 1313)$

$= (6866, 15)$

Step V. Calculation of secret-key by Alice is:

$K(x, y) = d_A * P_B$

$= 4701 * (6866, 15)$

$= (1846, 3967)$

Step VI. Calculation of secret-key by Bob is:

$K(x, y) = d_B * P_A$

$= 4802 * (7926, 5458)$

$= (1846, 3967)$

In this way, both parties get same secret key i.e. K(x, y) = (1846, 3967). The variable '*h'* gets rounded value of 1% of K(x) = 18.

Encryption of plain message by Alice (Sender)

Steps for encryption

Step I. Alice generates plain message as: '32148765'

Step II. Encoding:

Alice encodes the plain message into encoded message points in the elliptic curve as shown in **Table 2** and in the **Figure 2.**

Step III. Encryption:

Alice encrypts the encoded message points into cipher message points as shown in **Table 3** and in the **Figure 3** and send the same to Alice.

Here the message is passed to do encryption using ECC, which uses public key of receiver.

Decryption of cipher message points by Bob (receiver)

Steps for decryption of cipher message points

Step I. Decryption: Bob decrypts cipher message points into encoded message points as shown as in **Table 2** and in the **Figure 2**.

Step II. Decoding: Bob decodes the encoded points into plain message.

Step III. Bob gets plain message as: '32148765'.

Table 2: Plain points in the elliptic curve

Pmsg(X)	Pmsg(Y)
55	3252
20	2119
38	3336
74	3399
146	3323
128	3151
110	37
93	1787

Table 3: Cipher points in the elliptic curve

Cmsg(X)	Cmsg(y)
2716	8156
2729	736
2606	515
5065	1924
4675	7807
1806	6837
3427	896
6647	6331

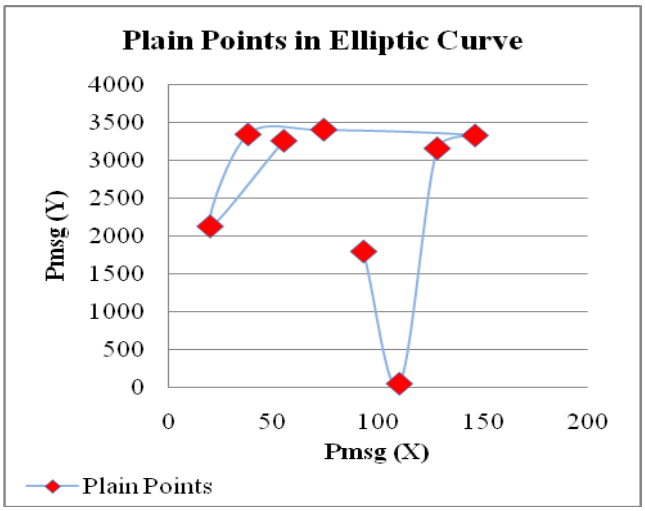

Figure 2: Plain points

Different Case Studies of implementation of RSA or/and ECC in S/W Security, H/W Security, Wireless LAN Security

Implementing Software Security

Public-key cryptography provides two important services of information security. They are as follows:

(i) Secrecy of information: It is provided by using encryption and decryption.

(ii) Authentication of information: It is provided by implementing a digital signature.

Secrecy of Information

Case Study 1: Comparative Analysis of Public-Key Encryption Schemes by BK Alese et al. [7]

This research work focuses on the comparative analysis of RSA encryption algorithm, ElGamal Elliptic Curve Encryption algorithm, and Menezes-Vanstone Elliptic Curve Encryption algorithm. These elliptic curves analog of ElGamal encryption scheme were implemented in Java, using

classes from the FlexiProvider library of ECC. The RSA algorithm used in the comparison is the FlexiProvider implementation. Performance evaluation of the three algorithms based on the time lapse for their key generation, encryption, and decryption algorithms, and encrypted data size was carried out and compared. The results show that their elliptic curve-based implementations are more superior to the RSA algorithm on all corresponding parameters.

After comparing the RSA and ECC ciphers, it was proved that ECC involves much fewer overheads compared to RSA. The ECC has shown to have many advantages due to its ability to provide the same level of security as RSA yet using shorter keys. However, its disadvantage which may even hide its attractiveness is its lack of maturity, as mathematicians, believed that enough research has not yet been done in ECDLP.

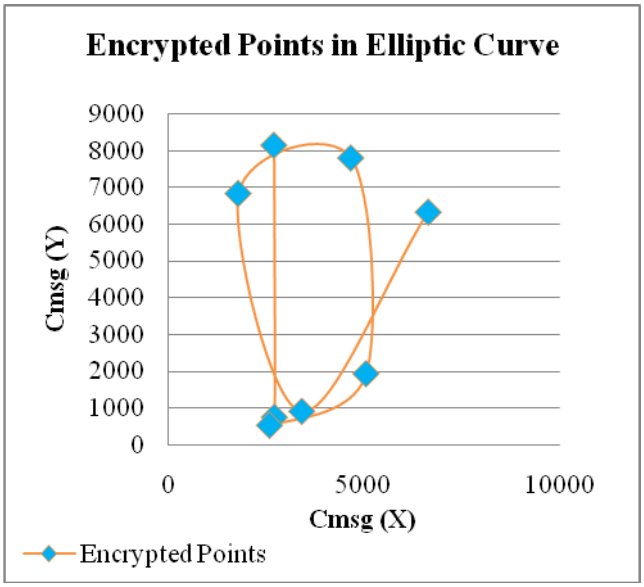

Figure 3: Cipher points

Authentication of Information

Case Study 1: Performance Comparison of Elliptic Curve and RSA Digital Signatures by Nicholas Jansma et al. [6]

This paper compares the performance characteristics of two public key cryptosystems (RSA and ECC) used in digital signatures to determine the applicability of each in modern technological devices and protocols that use such signatures.

Their findings suggest that RSA key generation is significantly slower than ECC key generation for RSA key of sizes 1024 bits and greater. RSA is comparable to ECC for digital signature creation regarding time and is faster than ECC for digital signature verification. Thus, for applications requiring message verification more often than a signature generation, RSA may be the better choice.

Case Study 2: A Secure and Efficient Remote User Authentication Scheme for Multi-server Environments Using ECC by Zhang, Junsong, et al. [13]

The requirements of operations are lesser in ECC-based than other related asymmetric-key schemes, in turn, it requires a less computational cost. The demonstration of the paper shows that proposed scheme can solve various types of security problems and is better suitable for memory-constrained devices.

Implementing Hardware Security

Case Study 1: Elliptic Curve Cryptosystems by M.J.B. Robshaw et al. [14]

In their note, they provide a high-level comparison of the RSA public-key cryptosystem and proposals for public-key cryptography based on elliptic curves.

There are, however, many issues to consider when making a choice between applications based on an elliptic curve cryptosystem and one based on RSA. In their note, they have presented some of the issues (security, performance, standards, and interoperability) that are perhaps most pertinent when making such a choice. The comparisons in this note are made, however, under the premise that an elliptic curve cryptosystem over GF(2^{160}) offers the same security as 1024-bit RSA.

Case Study 2: Comparing Elliptic Curve Cryptography and RSA on 8-Bit CPUs by Gura, Nils, et al. [3]

They propose a new algorithm to reduce the number of memory accesses. Implementation and analysis led to three observations: 1. Public-key cryptography is viable on small devices without hardware acceleration. On an Atmel ATmega128 at 8 MHz they measured 0.81s for 160-bit ECC point multiplication and 0.43s for an RSA-1024 operation with exponent e=2^{16}+1. 2. The relative performance advantage of ECC point multiplication over RSA modular exponentiation increases with the decrease in processor word size and the increase in key size. 3. Elliptic curves over fields using pseudo-Mersenne primes as standardized by NIST and SECG allow for high-performance implementations and show no performance disadvantage over optimal extension fields or prime fields explicitly selected for a particular processor architecture.

They compared elliptic curve point multiplication over three SECG/NIST curves secp160r1, secp192r1, and secp224r1 with RSA-1024 and RSA-2048 on two 8-bit processor architectures. On both platforms, ECC-160 point multiplication outperforms the RSA-1024 private-key operation by order of magnitude and within a factor of 2 of the RSA-1024 public-key operation. They presented a novel multiplication algorithm that significantly reduces the number

of memory accesses. This algorithm led to a 25% performance increase for ECC point multiplication on the Atmel AVR platform. Their measurements and analysis led to fundamental observations: The relative performance of ECC over RSA increases as the word size of the processor decreases. It stems from the fact that the complexity of addition, subtraction and optimized reduction based on sparse pseudo-Mersenne primes grows linearly with the decrease of the word size whereas Montgomery reduction grows quadratically. As a result, ECC point multiplication on small devices becomes comparable in performance to RSA public-key operations, and they expect it to be higher for large key sizes.

Wireless LAN Security

Case Study 1: Comparative Performance Analysis of Public-Key Cryptographic Operations in the WTLS Handshake Protocol by Rodríguez-Henríquez et al. [15]

In their paper, an efficient realization of the WTLS (Wireless Transport Layer Security) handshake protocol was implemented on a realistic wireless scenario composed of a typical mobile device wirelessly connected with a workstation server. The data gathered in their experiments shows that ECC consistently outperforms the traditional option represented by RSA in all the scenarios tested. Additionally, their analytical model predictions show a reasonable agreement with the obtained real data. They proposed a model for the protocol analysis considering the processing time of the cryptographic operations performed in the Client and the Server during the Negotiation protocol.

Comparative Analysis of RSA and ECC

This paper implements RSA and ECC for secrecy of information with three sample data inputs of 8 bits, 64 bits, 256 bits and random private keys based on the recommendation of NIST [2]. The experiments are done on MATLAB R2008a on Intel Pentium dual-core processor (1.60 GHz, 533 MHz, 1 MB L2 cache) with 2GB DDR2 RAM under Ms-Windows platform. The efficiency of ECC over RSA is shown in Table 4-6 and Figure 4-12. Based on experimentation, it is observed that RSA is very efficient in encryption but slow in decryption while ECC is slow in encryption but very efficient in decryption. Overall ECC is more efficient and secure than RSA as shown in the figures Figure [6, 9 and 12].

Table 4: 8 bits – Encryption, Decryption and Total Time (in seconds)

Input: 8 bits						
Security Bit Level	**Encryption**		**Decryption**		**Total Time**	
	ECC Enc. Time	RSA Enc. Time	ECC Dec. Time	RSA Dec. Time	ECC Total Time	RSA Total Time
80	0.4885	0.0307	1.3267	0.7543	1.8152	0.7850
112	2.2030	0.0299	1.5863	2.7075	3.7893	2.7375
128	3.8763	0.0305	1.7690	6.9409	5.6453	6.9714
144	4.7266	0.0489	2.0022	13.6472	6.7288	13.6962

Table 5: 64 bits – Encryption, Decryption and Total Time (in seconds)

Input: 64 bits						
Security Bit Level	**Encryption**		**Decryption**		**Total Time**	
	ECC Enc. Time	RSA Enc. Time	ECC Dec. Time	RSA Dec. Time	ECC Total Time	RSA Total Time
80	2.1685	0.1366	5.9099	5.5372	8.0784	5.6738
112	9.9855	0.1635	6.9333	20.4108	16.9188	20.5743
128	15.0882	0.1672	7.3584	46.4782	22.4466	46.6454
144	20.2308	0.1385	8.4785	77.7642	28.7093	77.9027

Table 6: 256 bits – Encryption, Decryption and Total Time (in seconds)

Input: 256 bits						
Security Bit Level	**Encryption**		**Decryption**		**Total Time**	
	ECC Enc. Time	RSA Enc. Time	ECC Dec. Time	RSA Dec. Time	ECC Total Time	RSA Total Time
80	7.9240	0.5596	22.8851	19.3177	30.8091	19.8772
112	39.7008	0.5815	26.3331	102.0337	66.0339	102.6153
128	58.4386	0.5611	27.4060	209.6086	85.8446	210.1697
144	77.5034	0.5718	32.1522	311.0649	109.6556	311.6368

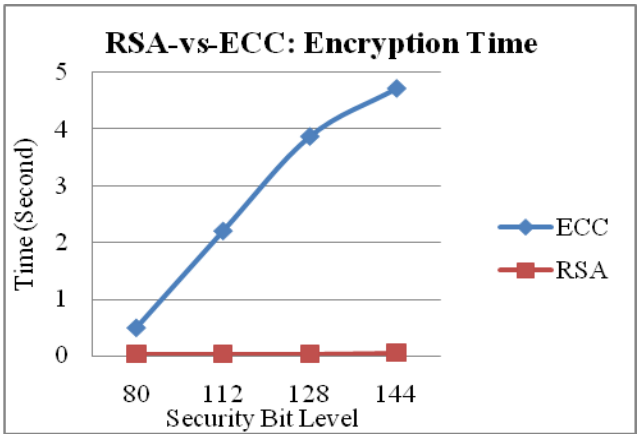

Figure 4: 8 bits – Encryption Time (in seconds)

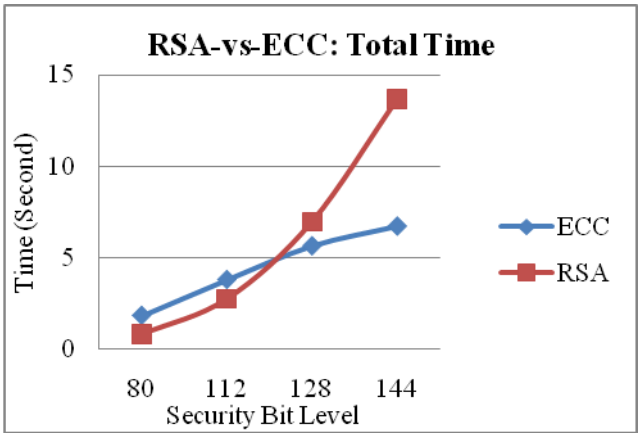

Figure 6: 8 bits – Total (Encryption and Decryption) Time (in seconds)

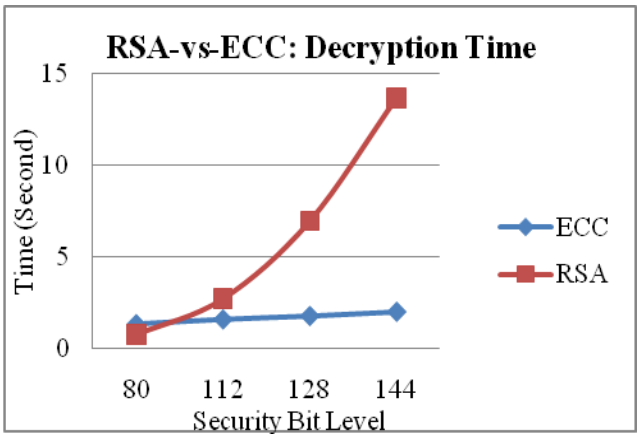

Figure 5: 8 bits – Decryption Time (in seconds)

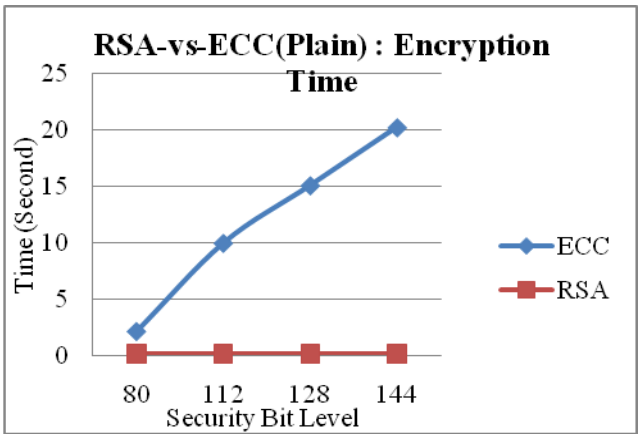

Figure 7: 64 bits - Encryption Time (in seconds)

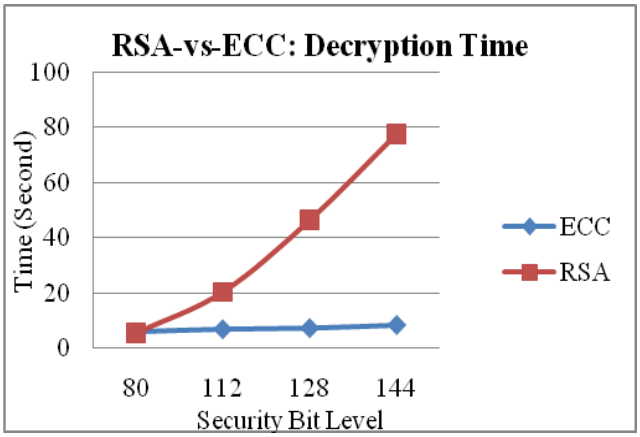

Figure 8: 64 bits - Decryption Time (in seconds)

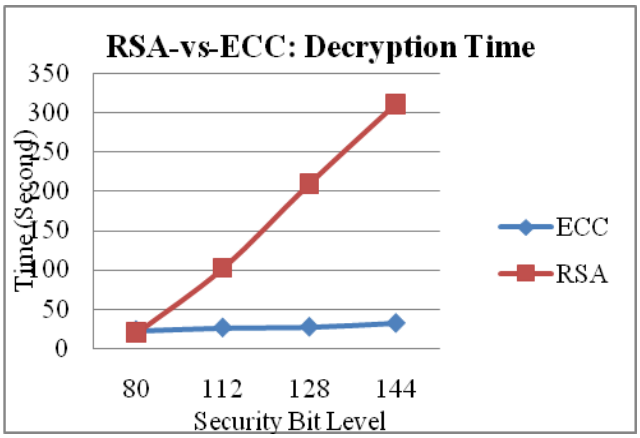

Figure 11: 256 bits – Decryption Time (in seconds)

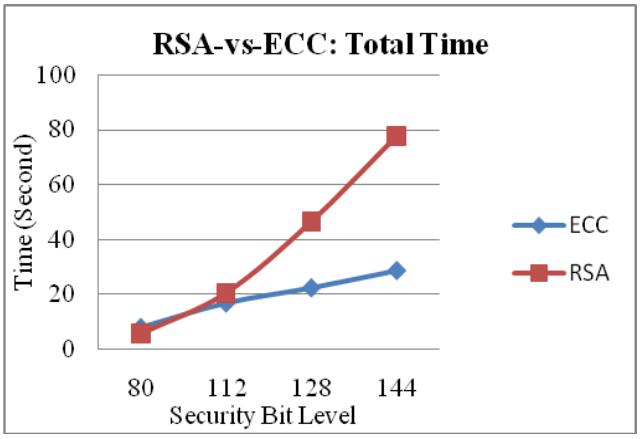

Figure 9: 64 bits - Total (Encryption and Decryption) Time (in seconds)

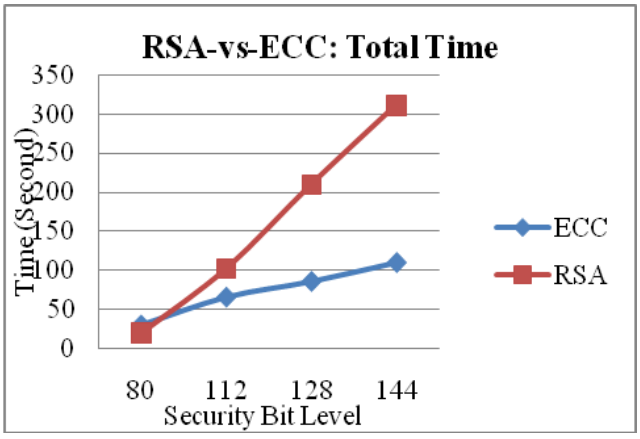

Figure 12: 256 bits – Total (Encryption and Decryption) Time (in seconds)

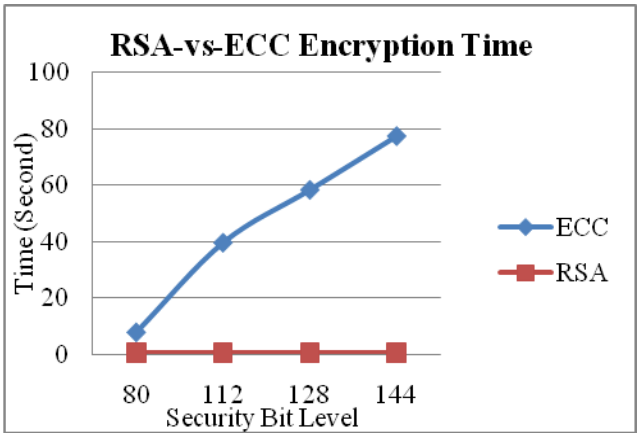

Figure 10: 256 bits – Encryption Time (in seconds)

CONCLUSION

Security of the message is paramount during its transmission from one user to another user or system. A cryptographic technique provides a message security. Symmetric-key cryptography is very good in providing security to the message but suffers from key distribution and management problems. To mitigate the key distribution and management problems and to ensure confidentiality, and integrity of a message, asymmetric-key cryptography has been invented by Diffie-Hellmen [12]. 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 devises.

ACKNOWLEDGEMENT

We would like to thank our colleagues, Head of Department of Computer Applications, Dean(R & C) and the Director of our Institute for supporting directly or indirectly in this research work.

REFERENCES

[1] Hankerson, D., Menezes, A.J. and Vanstone, S., 2006. *Guide to elliptic curve cryptography*. Springer Science & Business Media.

[2] Barker, E., Barker, W., Burr, W., Polk, W. and Smid, M., 2012. Recommendation for key management part 1: General (revision 3). *NIST special publication*, *800*(57), pp.1-147.

[3] Gura, N., Patel, A., Wander, A., Eberle, H. and Shantz, S.C., 2004, August. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *CHES* (Vol. 4, pp. 119-132).

[4] Bos, J., Kaihara, M., Kleinjung, T., Lenstra, A.K. and Montgomery, P.L., 2009. *On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography* (No. EPFL-REPORT-164549).

[5] Kute, V.B., Paradhi, P.R. and Bamnote, G.R., 2009. A software comparison of rsa and ecc. *Int. J. Comput. Sci. Appl*, *2*(1), pp.43-59.

[6] Jansma, N. and Arrendondo, B., 2004. Performance comparison of elliptic curve and rsa digital signatures. *nicj. net/files*.

[7] Alese, B.K., Philemon, E.D. and Falaki, S.O., 2012. Comparative analysis of public-key encryption schemes. *International Journal of Engineering and Technology*, *2*(9), pp.1552-1568.

[8] Mahto, D., Khan, D.A. and Yadav, D.K., 2016, June. Security Analysis of Elliptic Curve Cryptography and RSA. In *Proceedings of the World Congress on Engineering* (Vol. 1).

[9] Rivest, R.L., Shamir, A. and Adleman, L., 1978. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), pp.120-126.

[10] Miller, V.S., 1985, August. Use of elliptic curves in cryptography. In *Conference on the Theory and Application of Cryptographic Techniques* (pp. 417-426). Springer, Berlin, Heidelberg.

[11] Koblitz, N., 1987. Elliptic curve cryptosystems. *Mathematics of computation*, *48*(177), pp.203-209.

[12] Diffie, W. and Hellman, M., 1976. New directions in cryptography. *IEEE transactions on Information Theory*, *22*(6), pp.644-654.

[13] Zhang, J., Ma, J., Li, X. and Wang, W., 2014. A Secure and Efficient Remote User Authentication Scheme for Multi-server Environments Using ECC. *TIIS*, *8*(8), pp.2930-2947.

[14] Robshaw, M.J.B. and Yin, Y.L., 1997. Elliptic curve cryptosystems. *An RSA Laboratories Technical Note*, *1*, p.997.

[15] Rodríguez-Henríquez, F., López-Peza, C.E., León-Chávez, M.A. and Puebla, P., 2004, June. Comparative performance analysis of public-key cryptographic operations in the WTLS handshake protocol. In *Proceedings of the 1st International Conference on Electrical and Electronics Engineering* (pp. 24-27).

[16] Mahto, D. and Yadav, D.K., 2013, January. Network security using ECC with Biometric. In *International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness* (pp. 842-853). Springer, Berlin, Heidelberg.

[17] Mahto, D. and Yadav, D.K., 2015, February. Enhancing security of one-time password using Elliptic Curve Cryptography with biometrics for e-commerce applications. In *Computer, Communication, Control and Information Technology (C3IT), 2015 Third International Conference on* (pp. 1-6). IEEE.

[18] Mahto, D. and Yadav, D.K., 2015, March. Enhancing security of one-time password using Elliptic Curve Cryptography with finger-print biometric. In *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on* (pp. 1737-1742). IEEE.

[19] Mahto, D. and Yadav, D.K., 2016. Security Improvement of One-Time Password Using Crypto-Biometric Model. In *Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics* (pp. 347-353). Springer, New Delhi.

[20] Mahto, D. and Yadav, D.K., 2017. One-Time Password Communication Security Improvement using Elliptic Curve Cryptography with Iris Biometric. *International Journal of Applied Engineering Research*, *12*(18), pp.7105-7114.

[21] Mahto, D. and Yadav, D.K., 2017, Secure Online Medical Consultations Using Elliptic Curve Cryptography with Iris Biometric. *International Journal of Control Theory and Applications*, 10(13), pp.169-179.

[22] Verma, P., Mahto, D., Jha, S.K. and Yadav, D.K., Efficient RSA Cryptosystem with Key Generation using Matrix. *International Journal of Control Theory and Applications*, 10(13), pp.221-228.