https://doi.org/10.48009/2_iis_2017_103-112

Issues in Information Systems
Volume 18, Issue 2, pp. 103-112, 2017

IMPLEMENTING ELLIPTIC CURVE CRYPTOGRAPHY USING MICROSOFT EXCEL

Abhijit Sen, Kwantlen Polytechnic University, abhijit.sen@kpu.ca

ABSTRACT

Microsoft Excel offers a number of data manipulation tools that are extensively used to solve many business problems. However in Engineering and Computing Science these easy to use features are seldom used to solve appropriate problems. Instead most of the problems are solved using specific tools such as MATLAB.

The Rivest-Shamir-Adleman (RSA) algorithm is one of the most well-known and secure public-key encryption methods used for secured data transmission. However because of growth of mobile and wireless devices and the constraints of limited bandwidth, storage, and power associated with these devices Elliptic Curve Cryptography (ECC) is being increasingly adopted to provide adequate security with less computational resources. To determine the public key, and to encrypt and decrypt messages for ECC algorithm one has to use complex mathematical calculations. Many students of Introductory Computer Security courses without adequate mathematical background find it difficult to apply ECC algorithm to compute appropriate public keys and encrypt and decrypt messages. In this paper author attempts to show how the ECC algorithm can be modeled with Microsoft Excel, and how Excel can easily and effectively be used to encrypt and decrypt messages. The author uses several examples to show the viability of the usage of Excel to implement ECC algorithm. The author finds that the Excel can be easily used in the classroom to demonstrate how the ECC algorithm functions. The author also discusses the limitations and practical issues related to using Excel.

Keywords: Elliptic Curve Cryptography, ECC, Excel, Private Key, Public Key, RSA

INTRODUCTION

Spreadsheet software is one of the most widely used software in business today. Microsoft Excel is a powerful tool that has become entrenched in business processes to formulate models in business, to determine how limited resources can be utilized to achieve the major objective or stated goals of the systems. With its vast collection of mathematical formulas, and facilities available to create user-defined formulas, Excel is extensively used for data manipulations and visualization. Moreover Microsoft Excel is widely available on PCs and online and the application is easy to learn. In this paper the author shows how one can use Excel features to implement complex cryptographic algorithms, such as Elliptic curve cryptography (ECC).

ECC is an approach to public-key cryptography that utilizes algebraic structure of elliptic curves over finite fields. RSA, the popular cryptographic algorithm, requires long key size (1024- bits) and complex modulo operations which makes it unsuitable for resource constrained applications such as wireless sensor networks or mobile devices. ECC requires small key sizes (160- bits), and provides equal security for a smaller bit size, thereby reducing processing overhead (Ahmed, 2011), and (Amara, 2011). In mobile devices, one should not use resource intensive cryptography algorithms. In constrained environments, such as mobile devices, smart card, PDAs where computing power, storage and bandwidth are limited, ECC is being increasingly implemented as a popular cryptographic scheme. The objective of this paper is to show how one can use Excel to determine private and public keys, encrypt and decrypt messages using well known ECC algorithm. ECC uses public-key cryptographic algorithm involving two keys- public and private key. Public key is made public and is used for encrypting messages, whereas private key is held secret and is used for decrypting messages. Private and public keys are mathematically related. The key elements of ECC Algorithm are the following:

- Generation of private key given a public key for sender and receiver

- Encryption of the message by the sender
- Decryption of the message by the receiver

All the above steps will be executed using the chosen Elliptic curve. The operations using points on Elliptic curve are not very straight forward. Many students have difficulties in understanding the basis of ECC algorithm because of complexity of mathematics involved. The mathematics is especially daunting for students in Introductory Cryptography courses. In the pursuit of finding simpler ways of teaching the ECC algorithm, the author researched different software tools such as MATLAB and programming languages such as C#, Java, etc. Because of its simplicity and ease of use, Microsoft Excel is selected as a possible candidate for Cryptography courses. A brief introduction to Excel in class is sufficient to solve this problem. In this paper, the author has demonstrated with examples a way of using Microsoft Excel to generate public and private keys and to encrypt and decrypt messages using the principles of ECC.

LITERATURE SURVEY

The application of theory of Elliptic Curve to the Cryptography is extensively studied in literature. The suitability of applying ECC algorithms to mobile devices is discussed in Chou (2015), and Hamlin (2015). Many vendors have incorporated ECC algorithms in their products, and this number has only been on the rise. Discussions of ECC applications can be found in Chou (2015), and Amara (2011). Chou (2015) discussed the advantages of using ECC algorithm over RSA. The implementation of ECC algorithms can be found in Padma (2010), and Kodali (2012). To apply ECC as a public-key cryptographic scheme, the message must be represented as a point in Elliptic Curve. The mapping of a message as points in Elliptic curve is discussed in details in King (2009), and Verma (2013). The performance of elliptic curve cryptosystem heavily depends on an operation called point multiplication. The mathematics involved are discussed in details in Sawlikar (2012), Forouzan (2013), and Kodali (2012).

Microsoft Excel is used extensively in different business applications areas ranging from Marketing and Product Management, Human Resources Planning, Finance and Accounting to tasks such as manufacturing, transportation, shortest path determination, and task assignment. Because of the complexity of cryptographic algorithms, there are very few implementation of cryptographic algorithms using Excel. Most of these algorithms are implemented using high level languages such as Java, C++ etc.

In this paper the author investigates the use of Excel to implement ECC algorithm and demonstrates how to use Excel to formulate encryption and decryption algorithm. This tool is easy to use and will be suitable for users without substantial mathematical background to implement a solution.

OVERVIEW OF ECC ALGORITHM

ECC is a cryptosystem for public-key encryption used for securing sensitive data being transmitted over insecure channels such as the Internet (Forouzan, 2013), and (Stallings, 2017). In ECC two different keys are used, one public and the other private. The public key can be shared with everyone, whereas the private key must be kept secret. The most complex part of ECC cryptography is the public and the private key-generation algorithm, which is summarized below (Kodal, 2012), and (Verma, 2013).

- Select a suitable elliptic curve $E_p(a, b)$ and a point G on the elliptic curve, with parameters a, b, and p, where p is a prime number.
- Encoding a message - The message m to be encrypted is mapped as a point P_m on the elliptic curve
- ECC Key Generation Steps – generate private, public keys for both sender and receiver
- Encryption – Encrypt the message point P_m to generate cipher point C_m
- Decryption – Decrypt the message point C_m to get original point P_m
- Decoding a message point P_m to get the original message m

The following discusses the steps in details.

Select a suitable elliptic curve $E_p(a, b)$ and a point G Steps

- Select an elliptic curve of the form below:

  $(y^2) \bmod p = (x^3 + ax + b) \bmod p$, where a and b are real numbers, and p is a prime number

- Select base point G = (x1, y1) on Elliptic curve with large order n s.t. nG = O

Encoding a message Steps (King, 2009)

ECC Encryption and Decryption steps can only be applied to encrypt and decrypt a point on the elliptic curve. As such all the possible message (ASCII) values to be transmitted must be preprocessed to map them as points onto the elliptic curve.

Convert each character of the message to actual points on the elliptic curve.

- Take each of message characters and use it as the x coordinate for the point (x, y) on the elliptic curve.
- Compute y coordinate by substituting the value of x in the following equation:

  $(y^2) \bmod p = (x^3 + ax + b) \bmod p$, where a and b are real numbers, and p is a prime number.

ECC Key Generation Steps (Forouzan, 2013), (Kodali, 2012), (Padma, 2010)

Each user generates a public/private key using the following ECC algorithm steps:

- Sender A selects a private key $n_A < n$
- Receiver B selects a private key $n_B < n$
- Sender A computes public key, P_A, which is a point on the Elliptic curve (EC) using the equation: $P_A = n_A * G$.
- Sender B computes public key, P_B, which is a point on the Elliptic curve (EC) using the equation:
- $P_B = n_B * G$.
- Note: The attacker cannot easily compute private keys from Public keys alone due to the ECDLP's complexity.
- Sender A computes the shared key point using Receiver B's public key and his own private key using the following equation: $S_K = P_B * n_A = n_A * n_B * G$.
- Receiver B computes the shared key point using Sender A's public key and his own private key using the following equation: $S_K = P_A * n_B = n_A * n_B * G$.
- Note: Now both Sender A and Receiver B have the same shared secret key S_K which can be used to encrypt and decrypt any of the subsequent messages.
- Sender A , and Receiver B publish their public encryption key P_a, and P_b
- Sender A , and Receiver B keep secret shared key: S_K

Encryption Steps (Forouzan, 2013)

The encrypted text $Cm = \{K* G, P_{cm}\}$ is computed from the plaintext message M using the following steps:

- Sender A encodes or maps a message value M to a point, (P_m), on the EC.
- Sender A encrypts this plaintext message point (P_m), to obtain the corresponding cipher point, (P_C), lying on the same EC using the following equation: $P_{cm} = [P_m + K* P_b]$, where k is a random number chosen by Sender A
- Sender A computes the term K*G, where G is generator point on EC
- Sender A sends to Receiver B two cipher text points:

$Cm = \{K^* G, P_{cm}\}$

Decryption Steps (Forouzan, 2013)

Upon receiving the cipher point, $Cm = \{K^* G, P_{cm}\}$, Receiver B computes the following to decrypt the received cipher points Cm.

- Compute $S_K = K^*G^* n_B$
- Subtracts S_K the from the cipher point, (P_{cm}), to get the encoded message point (P_m), $P_m = [P_{cm} - S_K]$. Note: If (S_K) is represented by the point (x, y) on EC then the additive inverse ($-S_K$), is a point (x, p-y) on the same EC

Decoding a message Steps (Kodali**,** 2012)

The encrypted points are received at the receiving end and decrypted to get the original plaintext points which will then be converted to actual plaintext message.

- Receiver takes the x- coordinate of this decrypted plaintext message point, and computes (x-1)/K
- The quotient of this division operation corresponds to the ASCII value of the original message.

IMPLEMENTATION OF ECC

If one examines all the steps above, one can identify that the following operations are required [Kodali**,** 2012]:

- Addition of two points R = P + Q, where P is a point with coordinates (x_p, y_p), Q is a point with coordinates (x_q, y_q). R is a point coordinates (x_r, y_r). R = (x_r, y_r) is calculated using the following rules:

  $x_r = (\lambda^2 - x_p - x_{q)} \bmod p$ (1)

  $y_r = (\lambda\ (x_p - x_{r)} - y_{p)} \bmod p$ (2)

  Where

  $\lambda = ((y_q - y_p) / (x_q - x_p)) \bmod p$ if P <> Q (3)

  or

  $\lambda = ((3\ x_p^2 + a)/ (2\ y)) \bmod p$ if P = Q (4)

- Scalar multiplication of a point k*G where a scalar k multiplies the point G on the given elliptic curve with coordinates (x, y). This can be achieved by repeated addition of points operation.
- One also needs to compute modular inverse

SPREADSHEET MODELING OF ECC ALGORITHM

In this section, we show how the theoretical foundations of ECC algorithm as described in the previous section is translated to actual formulae so that problem can be implemented in Excel. The objective is to compute various keys required for encryption and decryption process and to encrypt and decrypt a message.

Table 1 summarizes the information required for the ECC algorithm. The first column, labeled "Public Information" lists the data made available to those who will take part in the communication and are not kept secret. The second column, labelled "Sender's Private Information", and the third column, labelled Receiver's Private Information are known only to sender and receiver respectively and must be kept secret and not divulged to any other parties. The fifth column, labelled "Sender-Receiver mutually agreed" defines a random constant K which both sender and receiver

must agree ahead of time.

The first column of Table 2, labelled Sender, lists the sequence of steps that sender must execute to encrypt a message destined for sender using ECC. The second column of Table 2, labelled Receiver, lists the sequence of steps that receiver must execute to decrypt the cipher text message from sender to get the original plaintext message.

Table 1. Summary of various parameters and keys for ECC algorithm

Public Information	**Sender's Private Information**	**Receiver's Private Information**	**Sender - Receiver mutually agreed**
Elliptic curve with parameters a, b, p	Private Key for Sender: N_a	Private Key for Receiver: N_b	Random integer K
Generator Point G on the Elliptic Curve			
Public Keys- for sender: P_a, for Receiver: P_b			

Table 2. Summary of Operations to be executed by Sender and Receiver

Sender	**Receiver**
1. Selects private key N_a, keeps it secret	1. Selects private key N_b, keeps it secret
2. Computes public key $P_a = N_a * G$, publishes P_a	2. Computes public key $P_b = N_b * G$, publishes P_b
3. Selects random integer K	3. Receives Pc from sender (see step 7 from sender)
4. Computes K*G	4. Compute $S_K = N_b * K*G$
5. Takes message MSG , maps a message value MSG to a point, on the EC (P_m) =(x,y)	5. Subtracts S_K the from the cipher point, (P_{cm}), to get the encoded message point (P_m), $P_m = [P_{cm} - S_K] =(x,y)$.
6. Encrypts plaintext message point (P_m) and computes cipher text : $P_{cm} = [P_m + K* P_b]$	6. Takes the x- coordinate of this decrypted plaintext P_m computes (x-1)/K to get original MSG
7. Sends to receiver two cipher text points: Pc = {K*G , P_{cm} }	

The above cryptographic steps can be represented using the following spreadsheet model in Excel:
Cell B7: Message to be encrypted
Cell F7, G7: x, y values respectively of point corresponding to message in cell B7

Cell L7: Coefficient a of the chosen Elliptic Curve
Cell M7: Coefficient b of the chosen Elliptic Curve
Cell O7: value of the Prime number p

Cell B14: Chosen private key N_a of sender
Cell C14: Chosen private key N_b of receiver
Cell D14, Cell E14: x, y values respectively of generator point G on the Elliptic Curve

Cell F14, Cell G14: x, y values respectively of sender's public key P_a corresponding to operation $P_a = N_a * G$
Cell H14, Cell I14: x, y values respectively of receiver's public key P_b corresponding to operation $P_b = N_b * G$

Cell B22: value of agreed upon random integer K
Cell D22, Cell E22: x, y values respectively corresponding to operation K* P_b

Cell G22, Cell H22: x, y values respectively corresponding to operation K*G
Cell I22, Cell J22: x, y values respectively corresponding to operation $P_{cm} = [P_m + K* P_b]$

Cell B31, Cell C31: x, y values respectively corresponding to received $P_{cm} = [P_m + K* P_b]$
Cell E31, Cell F31: x, y values respectively corresponding to received K*G
Cell H31, Cell I31: x, y values respectively corresponding to operations $S_K = N_{b*} K*G$
Cell J31, Cell KI31: x, y values respectively corresponding to operations - S_K

Cell M31, Cell NI31: P_m = x, y values respectively corresponding to operations $P_{m=} [P_{cm} - S_K]$
Cell P31: Decode Pm to get back plaintext

In order to implement ECC algorithms, the following user-defined functions are developed which will be applied to the appropriate cells above. These functions along with other built-in functions are used to implement ECC. In the following the elliptic curve parameters are a, and b. p is a chosen prime number.

Mapping message to points on Elliptic curve - The following two functions find the corresponding x, and y coordinates for a given message.

- Public Function MsgXvalue(Msg As Integer, k As Integer, a As Integer, b As Integer, p As Integer) As Integer - Returns x value corresponding to message Msg

- Public Function MsgYvalue(Msg As Integer, k As Integer, a As Integer, b As Integer, p As Integer) As Integer - Returns y value corresponding to message Msg

- Public Function MSqrt(a As Integer, b As Integer, p As Integer, Xj As Integer) As Integer - Finds a solution y to the Elliptic curve: $y^2 = x^3 + a*x + b$

Public Function modinv(number As Integer, p As Integer) As Integer – Returns modulus inverse of number

Compute P + Q - Find x and y values on Elliptic Curve for Z = P + Q where P=(x1,y1) and Q = (x2,y2)

- Public Function Xcordinate(a As Integer, x1 As Integer, y1 As Integer, x2 As Integer, y2 As Integer, p As Integer) As Integer – Returns x value for Z

- Public Function Ycordinate(a As Integer, x1 As Integer, y1 As Integer, x2 As Integer, y2 As Integer, p As Integer) As Integer - Returns y value for Z

- Public Function CalculateLambda(a As Integer, x1 As Integer, y1 As Integer, x2 As Integer, y2 As Integer, p As Integer) As Integer – Finds the value of Lambda needed to calculate x and y values on Elliptic Curve for P + Q where P=(x1,y1) and Q = (x2,y2)

- Public Function modinv(number As Integer, p As Integer) As Integer – Returns modulus inverse of number

Compute scalar multiplication n*G- Find x and y values on Elliptic Curve for Z = n*G where G=(x1,y1)

- Public Function nTimesGx(a As Integer, x1 As Integer, y1 As Integer, p As Integer, num As Integer) As Integer – Returns x value for Z

- Public Function nTimesGy(a As Integer, x1 As Integer, y1 As Integer, p As Integer, num As Integer) As Integer - – Returns x value for Z

Compute negative of G - Find x and y values on Elliptic Curve for Z = -G where G=(x1,y1)

- Public Function NegativeG(y As Integer, p As Integer) As Integer – Returns negative of G

The author demonstrates the viability of using Excel to implement ECC algorithm to encrypt and decrypt two messages: 7 and 9. Two different Elliptic Curves $\mathbf{y^2 = x^3 + 2x + 1}$ and $\mathbf{y^2 = x^3 - x + 1}$ are used for each message. The results are discussed in details below.

RESULTS

Figure 1, Figure 2, and Figure 3 display the formulation and computation of the ECC algorithm on an Excel spreadsheet. Figure 1 shows the chosen elliptic curve with parameters a=-1, b=1 and prime number is 127. The Generator point G is chosen to be (0, 1). The sender A uses the private key $N_a = 2$, and the receiver B uses the private key Nb = 3. Figure 2 shows the process of encryption. Figure 3 shows the result of decryption. It shows that the encrypted message is deciphered correctly to generate the original message 9.

Two test runs are executed using two different Elliptic Curves. Table 3 and Table 4 summarize results using the Elliptic Curve: $\mathbf{y^2 = x^3 + 2x + 1.}$ The prime number p=97 is used for this test. Two messages 7 and 9 are encrypted and as shown the two encrypted messages are decrypted correctly to get the original messages 7 and 9.

Table 5 and Table 6 summarize results using the Elliptic Curve: $\mathbf{y^2 = x^3 - x + 1.}$ The prime number p=127 is used for this test. Two messages 7 and 9 are encrypted and decrypted using this elliptic curve. As shown the two encrypted messages are decrypted correctly to get the original messages 7 and 9.

MSG	Pm		Elliptic Curve Parameters		Prime #
	X	Y	a	b	p
9	91	25	-1	1	127

Private Keys		Public Keys					
N_a	N_b	G		$P_a=N_a*G$		$P_b=N_b*G$	
		X	Y	X	Y	X	Y
2	3	0	1	32	15	56	38

Figure 1. Mapping message to points, computing Public keys

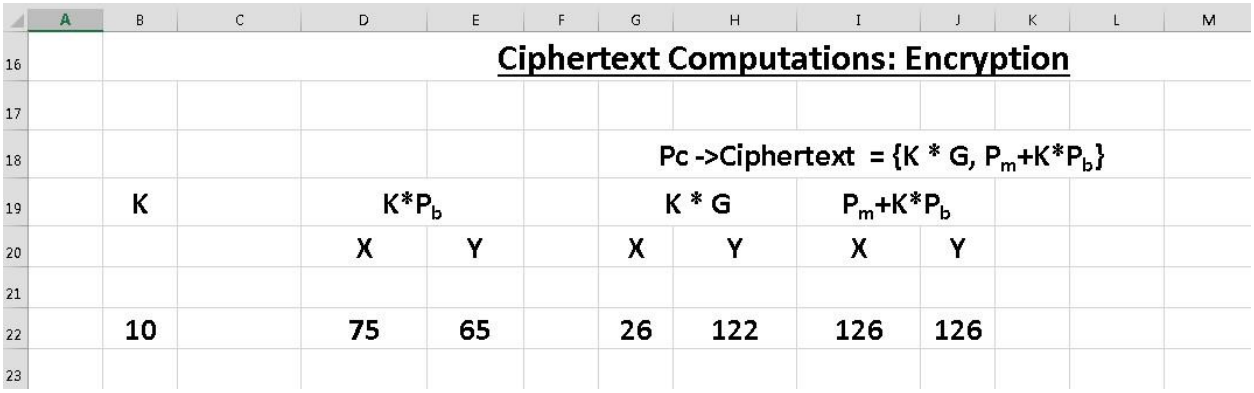

Figure 2. Encrypting message with a chosen value of integer K

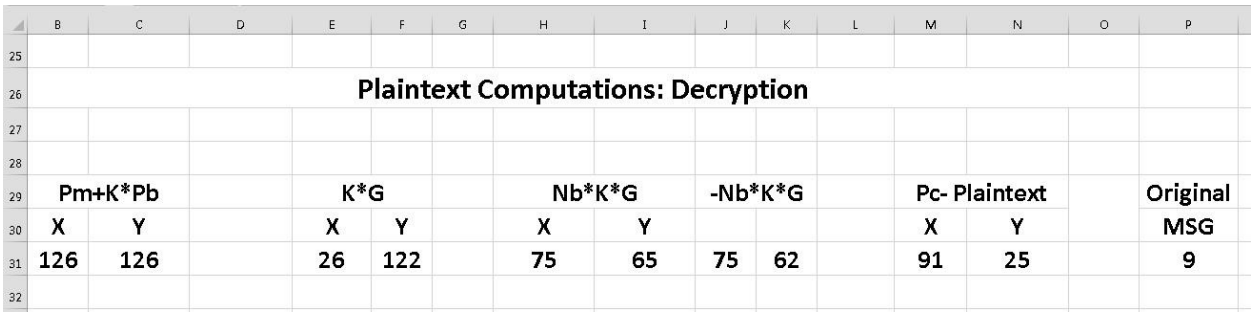

Figure 3. Decrypting cipher text points to plaintext points and decoding and plaintext points to original message

Table 3. Computation of Keys using Elliptic Curve: $y^2 = x^3 + 2x + 1$, G (0, 1), prime p = 97

Keys	**Sender (A)**	**Receiver (B)**
Private Key	N_a: 2	N_b: 3
Public Key	P_a: (1,95)	P_b: (8,23)

Table 4. Results of Encryption Decryption using Elliptic Curve: $y^2 = x^3 + 2x + 1$, G (0, 1), prime p = 97. Random number K =10

Message Data MSG	**Encoded Message P_m**	**Cipher Point P_c (After Encryption) { K * G, Pm+K*Pb}**	**Plaintext Point P_c (After Decryption)**	**Decoded Message**
7	(71,30)	{(62,30), (42,68) }	(71,30)	7
9	(90,41)	{(62,30), (63,60) }	(90,41)	9

Table 5. Computation of Keys using Elliptic Curve: $y^2 = x^3 - x + 1$, G (0, 1), prime p = 127

Keys	**Sender (A)**	**Receiver (B)**
Private Key	N_a: 2	N_b: 3
Public Key	P_a: (32,15)	P_b: (56,38)

Table 6. Results of Encryption Decryption using Elliptic Curve: $y^2 = x^3 - x + 1$, G (0, 1), prime p = 127. Random number K =10

Message Data MSG	**Encoded Message P_m**	**Cipher Point P_c (After Encryption) { K * G, Pm+K*Pb}**	**Plaintext Point P_c (After Decryption)**	**Decoded Message**
7	(70,44)	{(26,122), (86,92) }	(70,44)	7
9	(91,25)	{(26,122), (126,126) }	(91,25)	9

As can be seen from Table 3 and Table 5, using different Elliptic Curves result in different public keys although same private keys N_a: 2 , N_b: 3 are used in both test runs. The public keys for A: P_a= (1, 95) and for B: P_b= (8,23) are computed using Elliptic Curve: $y^2 = x^3 + 2x + 1$, whereas the public keys for A: P_a= (32,15) and for B: P_b= (56,38) are calculated using Elliptic Curve: $y^2 = x^3 - x + 1$.

For the same plaintext data the encoded message maps to different points on the elliptical curve. Message Data 7 maps to $P_{m=}$ (71, 30) for Elliptic Curve: $y^2 = x^3 + 2x + 1$ as shown in Table 4, whereas as shown in Table 6 same message data 7 maps to $P_{m=}$ (70,44)for Elliptic Curve: $y^2 = x^3 - x + 1$. Message Data 9 maps to $P_{m=}$ (90,41) for Elliptic Curve: $y^2 = x^3 + 2x + 1$ as shown in Table 4, whereas as shown in Table 6 same message data 9 maps to $P_{m=}$ (91,25)for Elliptic Curve: $y^2 = x^3 - x + 1$.

Similarly one can observe from Table 4 and Table 6 using different Elliptic Curves result in different encrypted messages for the same plaintext. Encrypted Message 7 maps to {(62, 30), (42, 68)}, and encrypted message 9 maps to {(62, 30), (63, 60)} when Elliptic Curve: $y^2 = x^3 + 2x + 1$ is used. Same Encrypted Message 7, and 9 map to {(26,122), (86, 92)}, and {(26,122), (126,126)} respectively when Elliptic Curve: $y^2 = x^3 - x + 1$ is used.
These examples demonstrate the viability of using Excel to implement the ECC algorithm.

LIMITATIONS OF EXCEL

For real life applications which require very large keys, Excel cannot be effectively and easily used. It is practically impossible to implement the ECC algorithm in a spreadsheet using very large integer values**.** Many built-in formulae such as Excel's mod(x, n) does not work if x > 2^32 - 1 (i.e., longer than 32 bits), although one can write software routines to handle large integers [Microsoft, 2013 - 2016]. However Excel is a very simple tool that one can use in a classroom to demonstrate the principles and concepts of ECC using numbers that Excel can handle.

CONCLUSIONS

This paper discusses how some of the Microsoft Excel features, such as using formulas, and creating user defined functions, so widely used in business applications can easily be applied to solve Elliptical Curve Cryptographic problems. The author demonstrates Excel's modeling techniques to compute private and public keys and to encrypt and decrypt messages using ECC algorithm. Many students of introductory courses in Computer Security and Cryptography have difficulties in understanding cryptography concepts because of the mathematical complexity of these algorithms. Instructors teaching cryptography courses can use simple software like Microsoft Excel to build a model, and can easily explain the model and its results to students who struggle with complexities and rigors of underlying mathematical principles. The users do not have to have knowledge in sophisticated programs like MATLAB and other simulation software to solve many mathematically challenging cryptography problems. For general users interested in the applied aspects of cryptography it may be worthwhile to try out simple easily available tools to convey the concepts without using sophisticated applications or simulation languages, even though they may offer higher performance and flexibility.

REFERENCES

Amara, M., & Siad, A. (2011). *Elliptic curve cryptography and its applications*. International Conference on Signal Processing and their Applications (WOSSPA), 247-250.

Ahmed, M. H., Alam, S. W., Qureshi, N., & Baig. I. (2011). Security for WSN based on ECC. ICCNIT, 75-79.

Forouzan, B. A. (2013). *Cryptography and Network Security.* McGraw-Hill. Available: http://www.mheducation.com/highered/product.M0073376221.html

Chou, W. (2015). *Elliptic Curve Cryptography and Its Applications to Mobile Devices*. 1-23. Available from: http://honors.cs.umd.edu/reports/ECCpaper.pdf

Hamlin, A. (2015). *Overview of Elliptic Curve Cryptography on Mobile Devices*. 1-16. Available from: http://www.cs.tufts.edu/comp/116/archive/ahamlin.pdf

King, B. (2009). Mapping an Arbitrary Message to an Elliptic Curve when Defined over GF(2^n). *International Journal of Network Security, 8*(2), 169-176. Available: https://pdfs.semanticscholar.org/c64c/ad5f74d28542904925adb1b7019146d09903.pdf

Kodali, R. K., & Sarma, N. V. S. (2012). ECC Implementation using Koblitz's Encoding. Proc. Int. Conf. on Communication Engineering and Network Technologies, *CENT, 37*(2), 411-417.

Microsoft Documentation. (2016-2013). *Excel specifications and limits*. Available: https://support.office.com/en-us/article/Excel-specifications-and-limits-1672b34d-7043-467e-8e27-269d656771c3#ID0EBABAAA=Excel_2016-2013

Padma, Bh., Chandravathi, D., & Roja, P. P. (2010). Implementation of Elliptic Curve Cryptography using Koblitz's Method. *International Journal on Computer Science and Engineering*, 2(05), 1904-1907. Available: https://pdfs.semanticscholar.org/bc2f/1260888bfb5e83a8f14bfc296a9f3fa8c87f.pdf

Sawlikar, A. (2012). Point Multiplication Methods for Elliptic curve Cryptography. International Journal of Engineering and Innovative Technology (IJEIT), 1(1), 1-4. Available: https://www.idc-online.com/technical_references/pdfs/data_communications/Point%20Multiplication.pdf

Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice.* Pearson, 7th Edition. Available: https://www.pearsonhighered.com/program/Stallings-Cryptography-and-Network-Security-Principles-and-Practice-7th-Edition/PGM334401.html

Verma, K., & Agrawal, H. (2013). Elliptic curve cryptography using Koblitz encoding method. *International Journal of Advanced Scientific and Technical Research, 3*(3), 418-422. Available: http://www.rspublication.com/ijst/index.html