Issues in Information Systems
Volume 24, Issue xx, pp. 147-158, 2023

DOI: https://doi.org/10.48009/1_iis_2023_113

Elliptic curve cryptography: implementation using Google Apps Script (GAS)

Abhijit Sen, *Kwantlen Polytechnic University, abhijit.sen@kpu.ca*
Arnab Sen, AKQA, *USA, arnabsen@gmail.com*

Abstract

The Rivest-Shamir-Adleman (RSA) and Elliptic Curve Cryptography (ECC) algorithms are two well-known public-key encryption methods for transmitting data securely over an untrusted network. ECC is increasingly being adopted as an alternative to RSA to provide adequate security to systems and networks with less computational resources such as limited bandwidth, storage, and power. Because of the complexity of mathematics involve, it is a challenge to students and practitioners alike without adequate mathematical foundations to apply the ECC algorithm to compute appropriate public keys and encrypt and decrypt messages. In this paper the authors attempt to show how the ECC algorithm can be modeled with Google Sheets, and how Google App Scripts (GAS) can easily and effectively be used to encrypt and decrypt messages. The authors find that the ECC algorithm can be easily demonstrated to learners utilizing Google Sheets as the foundation. The authors also discuss the limitations and practical issues encountered when using Google Sheets and GAS.

Keywords: elliptic curve cryptography, ECC, Excel, GAS, Google Sheets, IoT, private key, public key, RSA

Introduction

Spreadsheet software is widely used software in organizations that allow users to work with data in a variety of ways to create budgets, forecasts, inventories, schedules, charts, graphs and many other data-based worksheets. Users can capture, display and manipulate data organized in rows and columns. Some examples of the most commonly used spreadsheet software are :

- Microsoft Excel: It is available as a part of Microsoft Office suite for Windows, macOS, Android and iOS. It is also available as part of the cloud-based subscription service Office 365. Visual Basic for Application (VBA) is used as a Scripting Language to extend functionality of Excel.

- Google Sheets: It is available as part of Google's web-based application suite, Google Workspace. Google Sheets is free and also available Android and iOS. Google Apps Script (GAS) is used as a Scripting Language to extend functionality of Google Sheets and is based on JavaScript 1.6.

Google Sheets and Google Application Scripts is cloud-based, can be used without any license fees, and is freely available. Moreover, Google Application Scripts provide an easy way to automate processes with Google Sheets and other Google products. It is also easy to learn.

Google Application Scripts can be used to implement Elliptic Curve Cryptography algorithms. The scripts can be used to generate cryptographic keys and perform encryption, decryption processes. Additionally, it can be used for signing, and verification operations. generating key exchange agreements and authenticating users .

ECC is a public-key cryptography algorithm that utilizes modular arithmetic of elliptic curves over finite fields, in contrast to another popular RSA algorithm which is based on the complexity of factoring large numbers. ECC provides the same level of security as in RSA with smaller key sizes thereby shortening processing overheads (Ahmed, 2011), and (Amara, 2011). As a result, ECC is increasingly used in resource-constrained environments with limited computing storage and processing power such as mobile devices.

The main objective of the research is to show how one can use cloud-based Google Sheets features and GAS to implement complex cryptographic algorithms, such as Elliptic Curve Cryptography (ECC). Many users have difficulties in understanding the basis of ECC algorithm because of the complexity of mathematics involved. The authors investigated different software tools available for implementing ECC algorithms. The ECC algorithm has been implemented in many programming environments including MATLAB (Silverwood, 2007), Java (Martínez et al., 2010), Python (Bray, 2020), and Rust (Lovecruft et al., 2017) to name a few. To implement these tools, one needs some knowledge of the appropriate programming language. For novice users without coding experience these implementations are difficult to use.

One of the difficulties practicing or new security professionals or students face is their limited exposure to complex mathematical structures required to grasp the working steps of the ECC algorithm and implement encryption and decryption processes correctly. Google Sheets and Google Application Scripts can be used to teach the ECC algorithm to students and professionals. Moreover, users do not need to have a coding background to use the system. An ancillary benefit is that it is also a great way to introduce them to coding, as the scripts are easy to write and debug.

Literature Review

The fundamental concepts of mathematical theory and relevant algorithms of Elliptic Curve Cryptosystems have been thoroughly discussed and explained in various books authored by Hankerson (2006), Schneier (2015), and Stallings (2017). The performance of the Elliptic Curve algorithm is largely dependent on the efficient implementation of arithmetic of elliptic curves over finite fields. The mathematics involved and their implementations are discussed in detail in Kodali (2012), Sawlikar (2012), and Forouzan (2013). The mathematics needed to implement Elliptic Curve Cryptography (ECC) has been critically analyzed by Rabah (2005), and Cohen et al. (2012).

The advantages and challenges of ECC over other cryptosystems, such as efficiency and the size of the key needed to attain a certain level of security are also discussed by Rabah (2005). Bao (2022) analyzes the performance comparison of ECC and RSA using 3 samples of input data based on NIST-recommended 64-bit, 8-bit, 256-bit random keys. Ullah et al. (2023) in their study, concluded that the adoption of ECC methods provides significant benefits in distributed computing and heterogeneous networking and that ECC can be effectively used in cloud computing, e-health, and e-voting contexts, in addition to other common applications.

ECC algorithms have been implemented by many vendors in their products. ECC algorithms have widespread applications in mobile devices, Internet of Things (IoT) and blockchain technologies. The

suitability of applying ECC algorithms to mobile devices is discussed in Chou (2015), and Hamlin (2015). Pinol et al. (2015) implemented a lightweight BSD-licensed ECC system to provide security for the IoT devices and carried out thorough performance analysis using several other implementations and optimization algorithms. Umucu (2022) gave an overview on how Blockchain uses ECC encryption/decryption algorithms for securing transactions, and digital signatures for authenticating transactions. Yadav (2021) elaborated on the suitability of ECC in blockchain based IoT applications by providing a comparative analysis of the RSA and ECC algorithms. Saqan (2022) has explained the use of ECC to generate digital signatures for Bitcoin transactions.

Google Sheets is a popular cloud-based spreadsheet application and interface that lets users create and format spreadsheets directly using a web browser, and no additional software is required. Google Sheets has many functionalities similar to what Microsoft Excel offers. Google Apps Script, extends the functionalities of Google Sheets and can automate tasks by creating custom scripts for manipulating data and for connecting them with other applications.

Because of the mathematical complexity of ECC algorithms, users without substantial mathematical background encounter many challenges to implement and use the algorithm. In this research, the authors investigate the use of Google Sheets to model the ECC algorithm and GAS to implement the algorithm to demonstrate encryption and decryption process. The implementation is easy to use and is suitable for users without sufficient mathematical background to implement a solution. Moreover, there are plenty of tutorials and articles available for both novice users and experts to learn and use these tools (Ferreira, 2014; Ganapathy, 2016; Mcpherson, 2016).

The rest of the paper is organized as per the following manner. The ECC algorithm steps are described in the section titled "ECC Algorithm – Brief Overview". The section titled "ECC Implementation using Google Sheet and Google Application Script " discusses the implementation mechanisms for encryption/decryption process using Google Sheets and GAS. The output of important steps of ECC algorithm is shown in the section titled "Result" followed by conclusion in the section titled "Conclusions". Figures for implementation are provided in the Appendix A shows the list of abbreviations used and Appendix B contains the consolidated GSA script prototypes for ease of reference.

ECC Algorithm – Brief Overview

ECC is a public-key cryptosystem and may be used to transmit data securely over insecure channels or media, such as the Internet. Like any public key system, ECC uses two different keys - one public and the other private. The public key can be shared with everyone, whereas the private key must be kept secret. In ECC, the public and private keys are generated using the mathematical properties of elliptical curves. It is easy to compute a public key given a private key but it is computationally difficult to recover a private key from the knowledge of public key using the elliptic curve discrete logarithm function.

To use the ECC cryptography system the following major steps are required, which are listed in Figure 1 and Figure 2:

a. Choose a suitable elliptic curve Ep (a, b) and a point G on the elliptic curve, where a and b are real numbers, and p is a prime number.

- Ep (a, b) is in the form:
  $(y^2) \bmod p = (x^3 + ax + b) \bmod p$, where a and b are real numbers, and p is a prime number.
- Base point G = (x1, y1) is then selected on the elliptic curve with large order n s.t. nG = O

b. Encode plaintext message m by mapping m as a point Pm on the elliptic curve (King, 2009). This is done by:
   - Taking each of message characters and use it as the x coordinate for the point (x, y) on the elliptic curve
   - Calculate y coordinate by substituting the value of x in the elliptic curve Ep (a, b) chosen in step a above:
     $(y^2) \bmod p = (x^3 + ax + b) \bmod p$

   The computation is shown in Step 5 of Figure 1.

c. Generation of ECC Keys/Encryption and Decryption of message
   These steps are shown for both sender and receiver in details:

   - ECC Keys Generation –private, public keys for both Sender A and Receiver B are created (Kodal, 2012), and (Sawlikar, 2012). These are shown in Steps 1 – 4 of Figure 1, Figure 2. for both Sender A and Receiver B

   - Encryption of message point Pm - to generate compute cipher point Cm (Forouzan, 2013) is done by Sender A. This is shown in Steps 6- 7 of the Sender A of Figure 1.

   - Decryption of the message point Cm to get the original point Pm (Forouzan, 2013) is done by Receiver B. This is shown in Step 5 of the Receiver B of Figure 2.

   - Decoding the message point Pm to get the original message m (Kodali, 2012) is done by Receiver B. This is shown in Step 6 of the Receiver B of Figure 2.

Sender A	
1.	Selects private key N_a, keeps it secret
2.	Computes public key $P_a = N_a * G$, publishes P_a
3.	Selects random integer K
4.	Computes K*G, and $S_K = N_a * K*G$
5.	Takes message MSG, maps a message value MSG to a point, on the EC (P_m) =(x,y)
6.	Encrypts plaintext message point (P_m) and computes cipher text: $P_{cm} = [P_m + K* P_b]$
7.	Sends to receiver two cipher text points: Pc = {K*G , P_{cm} }

Figure 1: Steps to be executed by Sender A

Receiver B	
1.	Selects private key N_b, keeps it secret
2.	Computes public key $P_b = N_b * G$, publishes P_b
3.	Receives Pc from sender (see step 7 from sender)
4.	Compute $S_K = N_b * K*G$
5.	Subtracts S_K the from the cipher point, (Pcm), to get the encoded message point (Pm), Pm = [Pcm - S_K] =(x,y).
6.	Takes the x- coordinate of this decrypted plaintext Pm computes (x-1)/K to get original MSG

Figure 2: Steps to be executed by Receiver B

ECC Implementation using Google Sheet and Google Application Script

In this section, we demonstrate how the steps of the ECC algorithm, discussed in previous section, can be implemented in Google Sheets using GAS to create user-defined functions for various computations required for the encryption and decryption process. The goal is to calculate the various keys necessary for

encryption and decryption, and to apply encryption and decryption to a message. If steps outlined in Figure 1, and Figure 2 are critically analyzed, it can be seen that following general operations are needed to execute ECC algorithm [Kodali, 2012]:

- Addition of two points P (x_p, y_p), Q (x_q, y_q) to generate a resulting point R (x_r, y_r): R = P + Q. P, Q and R are all points on elliptic curve, and R is a resulting point on elliptic curve (such as P_m + K* P_b). For discussion of the details of calculating R, one can refer to [Kodali, 2012].

- Scalar multiplication of the form: U * G where U is a scalar multiplier and G is a point on elliptic curve (such as Na * G, K*G..).
- - G: Negative Value of G where G is a point on an elliptic curve (such as - S_K in $P_m = [P_{cm} - S_K]$).

The steps outlined in Figure 1, and Figure 2, can be laid out in the form of a Google Sheets for computation as shown in Figure 3.

Google Sheet Set Up for Message Encoding & Key Generation Process	
Cell Reference	**Description**
A7	Message MSG to be encrypted
C7, D7	Mapping of message MSG in cell A7 to a point (x, y) on elliptic curve
E7, F7	Coefficient a, b of the chosen elliptic curve
G7	Value of the chosen prime number p
A14, B14	Private key N_a, N_b chosen by Sender A and Receiver B
C14, D 14	x, y coordinates of generator point G on the elliptic curve
E14, F 14	x, y coordinates of Sender A's public key Pa based on operation $P_a = N_a * G$
G14, H14	x, y coordinates of Receiver B's public key Pb based operation $P_b = N_b * G$
Google Sheet Set Up for Encryption Process	
Cell Reference	**Description**
A19	Random integer K mutually agreed upon by Sender A and Receiver B
C22, D22	x, y coordinates of resulting operation K* P_b
E22, F22	x, y coordinates of resulting operation K* G
G22, H22	x, y coordinates of resulting operation $P_{cm} = [P_m + K* P_b]$
Google Sheet Set Up for Decryption Process	
Cell Reference	**Description**
A33, B33	x, y coordinates of resulting operation $P_{cm} = [P_m + K* P_b]$
C33, D33	x, y coordinates of resulting operation K*G
E33, F33	x, y coordinates of resulting operation $S_K = N_b* K*G$
G33, H33	x, y coordinates of resulting operation – S_K = - N_b* K*G
D39, E39	x, y coordinates of resulting operation $Pm= [P_{cm} - S_K]$
F39	Plaintext message MSG retrieved by Receiver B after decoding P_m

Figure 3. Google Sheets modelling with cell reference

To implement the ECC algorithm, various scripts were designed and developed using GAS, as shown in Appendix B. These scripts were applied to the appropriate cells previously defined.

Results

The algorithm is tested using two different Elliptic Curves.
The following Elliptic Curves with prime number p = 127 and base point G (0, 1):

- $y^2 = x^3 + 2x + 1$
- $y^2 = x^3 - x + 1$.

are used for these tests. For our examples, the private and public keys used for both elliptic curves are shown in Table 1. Two different messages 5 and 4 are encrypted and decrypted using ECC. Figure 4, Figure 5, and Figure 6 show the results in Google Sheet of Message Encoding, Key Generation, Encryption and Decryption process when Sender A transmits message 5. The results are shown using Elliptic Curve: y2 = x3 +2x + 1, G (0, 1), prime p = 127 for encoding, key generation, encryption and decryption. As shown the encrypted message is decrypted correctly to get the original messages 5.

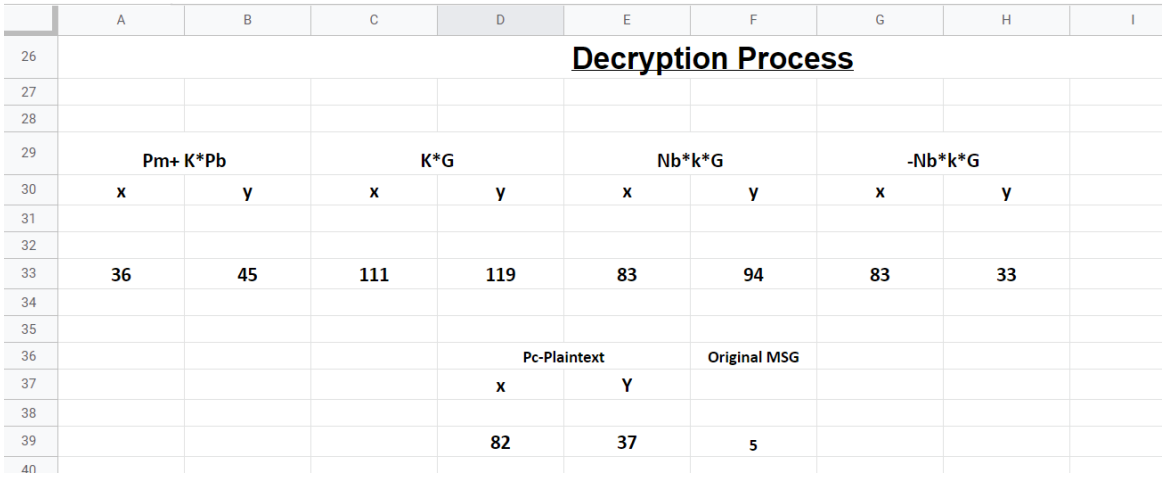
.

	A	B	C	D	E	F	G	H
1								
2								
3			**Message Encoding and Key Generation Process**					
4								
5	MSG		Pm		Elliptic Curve Parameters			
6			X	Y	a	b	Prime # p	
7	5		82	37	2	1	127	
8								
9	Private Keys				Public Keys			
10								
11	Na	Nb	G		Pa= Na*G		Pb=Nb*G	
12			x	y	x	y	x	y
13								
14	3	4	0	1	8	23	71	26
15								

Figure 4: Message Encoding and Key Generation Computations

	A	B	C	D	E	F	G	H	I
15									
16				**Encryption Process**					
17									
18									
19	K		K*Pb		K*G		Pm+K*Pb		
20			x	y	x	y	x	y	
21									
22	15		83	94	111	119	36	45	
23									
24									

Figure 5: Message Encryption Computations



	A	B	C	D	E	F	G	H	I
26				**Decryption Process**					
27									
28									
29	**Pm+ K*Pb**			**K*G**		**Nb*k*G**		**-Nb*k*G**	
30	**x**	**y**	**x**	**y**	**x**	**y**	**x**	**y**	
31									
32									
33	**36**	**45**	**111**	**119**	**83**	**94**	**83**	**33**	
34									
35									
36				**Pc-Plaintext**		**Original MSG**			
37				**x**	**Y**				
38									
39				**82**	**37**	**5**			
40									

Figure 6: **Message Decryption Computations**

The same test is repeated for message 4 using another Elliptic Curve: $y^2 = x^3 - x + 1$, G (0, 1), prime p = 127. The results of the two tests are summarized in Tables 2,3.

Table 1. **Private and Public Keys used for both Elliptic Curves: $y^2 = x^3 + 2x + 1$, and $y^2 = x^3 - x + 1$, G (0, 1), prime p = 127**

Key Type	Sender A	Receiver B
Private key	Na: 3	Nb : 4
Public Key	P_a : (8,23)	Pb : (71,26)

Table 2 summarizes the results using two different values of K for messages 5, 4 using Elliptic Curve: $y^2 = x^3 + 2x + 1$, G (0, 1), prime p = 127.

Table 3 summarizes the results using two different values of K for messages 5, 4 using Elliptic Curve: $y^2 = x^3 - x + 1$, G (0, 1), prime p = 127.

Table 2. **Results of Encryption and Description using Elliptic Curve: $y^2 = x^3 + 2x + 1$, G (0, 1), prime p = 127**

Value of K	Original Message MSG	MSG -> Encoded Message Pm : (x,y)	Cipher Point After Encryption Pc: (x,y) { K * G, Pm + K*P_b}	Pc ->Plaintext Point (After Decryption)	Decoded Message
15	**5**	**(82, 37)**	**{(111, 119), (36,45)}**	**(82, 37)**	**5**
	4	**(60,27)**	**{(111, 119), (101,91)}**	**(60,27)**	**4**
20	**5**	**(100, 40)**	**{(57, 74), (3,65)}**	**(100, 40))**	**5**
	4	**(82,37)**	**{(57, 74), (84,52)}**	**(82,37)**	**4**

Table 3. Results of Encryption and Description using Elliptic Curve: $y^2 = x^3 - x + 1$, G (0, 1), prime p = 127

Value of K	Original Message MSG	MSG -> Encoded Message Pm : (x,y)	Cipher Point After Encryption Pc: (x,y) { K * G, Pm + K*P_b}	Pc ->Plaintext Point (After Decryption)	Decoded Message
15	5	(75, 62)	{(47, 79), (75,65)}	(75, 62)	5
	4	(60,26)	{(47, 79), (70,44)}	(60,26)	4
20	5	(100, 41)	{(107, 28), (103,67)}	(100, 41)	5
	4	(81,45)	{(107, 28), (114,22)}	(81,45)	4

Examining the tabulated results in Table one can observe the following:

- The public keys for A: P_a= (8, 23) and for B: P_b= (71,26) are computed using Elliptic Curve: $y^2 = x^3 + 2x + 1$, whereas the public keys for A: Pa= (8, 23) and for B: P_b = (71, 26) are calculated using Elliptic Curve: $y^2 = x^3 - x + 1$. They are the same as they both use the same G (0,1), same Na: 3, N_b: 4 values, same p =127 for both elliptic curves.

- The encoded message for the same plaintext data maps to different points on the elliptical curve if the elliptic curve is changed. Message Data 5 maps to P_m = **(82, 37)** for Elliptic Curve: $y^2 = x^3 + 2x + 1$ as shown in Figure 4, whereas as shown in Figure 5 same message data 5 maps to P_m= **(75, 62)** for Elliptic Curve: $y^2 = x^3 - x + 1$ for the same value of K = 15. However, if K is changed to 20, Pm changes to **(100, 40)** for Elliptic Curve: $y^2 = x^3 + 2x + 1$, and P_m = **(100, 41)** for Elliptic Curve: $y^2 = x^3 - x + 1$. Similar observations can be seen if K is changed to 20.

- Similarly, one can observe from Table 2 and Table 3 using different Elliptic Curves result in different encrypted messages for the same plaintext message 5. Encrypted Message P_c maps to {**(111, 119), (36,45)}**, when Elliptic Curve: $y^2 = x^3 + 2x + 1$ is used, and P_c maps to {**(47, 79), (75,65)},** respectively when Elliptic Curve: $y^2 = x^3 - x + 1$ is used for the same value of K = 15. Similar observations can be seen if K is changed to 20.

These examples highlight the simplicity of implementing the ECC algorithm using Google Sheets with the help of custom functions created through Google Application Scripts. Google Apps Script is free to use and eliminates the need for expensive software licenses or server costs, or any need for users to set up and maintain their own server infrastructure. Google Apps Script run on Google's cloud infrastructure which ensures the availability and reliability of applications.

However, Google Apps Script has strict execution limits that restrict the amount of processing time and API requests per day. This can limit the scale of the applications that can be built using Google Apps Script. It is also important to note that while GAS and Google Sheets can handle large integers, the execution speed of GAS scripts is limited, with a maximum runtime of 6 minutes for free accounts before it terminates abruptly. Moreover, Google Apps Script is not designed for heavy-duty processing and may not perform optimally for large datasets or complex computations.

Discussions & Conclusions

The use of cryptography is crucial in securing data, and the Elliptic Curve Cryptography (ECC) algorithm is a widely adopted public-key encryption method. However, the complex mathematical concepts involved in ECC can make it challenging for students and novice users to understand and implement ECC. In this research, the authors present a novel approach to learning ECC using the Google Apps Script platform. Google Apps Script is a user-friendly platform for rapid application development that offers features such as custom functions and macros for Google Sheets.

The authors show how this platform can be used to solve ECC problems, making it easier for students and novice users to grasp the concepts of cryptography. The Google Sheets and Google Application Scripts approach is more accessible than sophisticated applications like MATLAB or Java-based tools, offering a scripting language that simplifies the process of learning and implementing ECC. Through the use of simple examples, the authors demonstrated how Google Sheets modeling techniques and GAS can be used to compute private and public keys and encrypt and decrypt messages using the ECC algorithm. This approach makes the process of learning ECC more accessible, especially for those without a strong mathematical background. The authors illustrated how the Google Apps Script platform can be used to learn and implement the Elliptic Curve Cryptography algorithm. With its user-friendly interface and scripting language, the Google Sheets and Google Application Scripts approach makes the process of learning ECC more accessible and manageable for students and novice users.

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.

Bao, J. (2022). Research on the Security of Elliptic Curve Cryptography. Advances in Economics, Business and Management Research, volume 652 Proceedings of the 2022 7th International Conference on Social Sciences and Economic Development (ICSSED 2022),

Bauer, W. (2002). Implementing Elliptic Curve Cryptography. Advanced Communications and Multimedia Security. IFIP — The International Federation for Information Processing, vol 100. Springer, Boston, MA.
Retrieved January 24, 2023 from https://doi.org/10.1007/978-0-387-35612-9_3

Bray , S. W. (2020). Implementing Cryptography Using Python, 2nd Edition, Wiley Media

Cohen, H, Frey., G., Avanzi, R., Doche, C., Lange, T., Nguyen, K., Vercauteren, F. (2012). Handbook of Elliptic and Hyperelliptic Curve Cryptography, 2nd Edition, Chapman & Hall/CRC

Ferreira , J. (2014) Google Apps Script: Web Application Development Essentials, 2nd Edition, O'Reilly Media

Ganapathy, R. (2016) Learning Google Apps Script: Customize and automate Google Applications using Apps Script, Packt Publishing

Hamish, S. (2007) A MATLAB implementation of elliptic curve cryptography. University of Canterbury. Mathematics and Statistics, Retrieved January 24, 2023 from https://ir.canterbury.ac.nz/handle/10092/10163

Hankerson, D., Menezes, A., Vanstone, S (2006). Guide to Elliptic Curve Cryptography. Springer New York, NY 1st edition.

Martınez, V. G., Encinas, L. H., Avila, C. S. (20101). A Java implementation of the Elliptic Curve Integrated Encryption Scheme. Retrieved January 24, 2023 from Consejo Superior de Investigaciones Cientificas

Lovecruft , I., de Valence, J. (2017). Fast, Safe, Pure-Rust Elliptic Curve Cryptography. Retrieved January 24, 2023 from https://tlu.tarilabs.com/images/cryptography/pure-rust-ecc/Fast%20Safe%20Pure-Rust%20Elliptic%20Curve%20Cryptography%20(Slides).pdf

Mcpherson , B. (2016) Going GAS: From VBA to Google Apps Script ,1st Edition, O'Reilly Media.

Pinol, O.P., Raza, S., Eriksson, J., & Voigt, T. (2015). BSD-based elliptic curve cryptography for the open Internet of Things. 2015 7th International Conference on New Technologies, Mobility and Security (NTMS), 1-5.

Rabah, K. , (2005). Theory and Implementation of Elliptic Curve Cryptography. Journal of Applied Sciences, 5: 604-633. Retrieved January 24, 2023 from https://scialert.net/fulltext/?doi=jas.2005.604.633

Saqan, S., (2022). Explanation of Bitcoin's Elliptic Curve Digital Signature Algorithm. Retrieved January 24, 2023 from https://suhailsaqan.medium.com/explanation-of-bitcoins-elliptic-curve-digital-signature-algorithm-6603f951863a

Schneier, B. (2015) Applied Cryptography: Protocols, Algorithms and Source Code in C, Wiley 1st edition.

Silverwood, H. (2007). A MATLAB implementation of elliptic curve cryptography (2007). University of Canterbury. Mathematics and Statistics, Retrieved January 24, 2023 from https://ir.canterbury.ac.nz/handle/10092/10163

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

Ullah,S. , Zhenga,J., Din,N., Hussain,.M., Ullah,F., Yousaf,M.(2023). Elliptic Curve Cryptography; Applications, challenges, recent advances, and future trends: A comprehensive survey, Computer Science Review, Volume 47,2023. Retrieved January 24, 2023 from https://www.sciencedirect.com/science/article/pii/S1574013722000648

Yadav, A. K.,(2021) Significance of Elliptic Curve Cryptography in Blockchain IoT with Comparative Analysis of RSA Algorithm, International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 2021, pp. 256-262, Retrieved January 24, 2023 from https://ieeexplore.ieee.org/document/9397166

Appendix A: List of Symbols Used

Symbols used	Description
m	Message to be encrypted
P_m	Message m is mapped to a point Pm on the elliptic curve
C_m	Message point Pm is encrypted to generate cipher point Cm
$E_p(a, b)$	Elliptic Curve ($\mathbf{y}^2$) mod p = (x^3+ ax + b) mod p with parameters a, b and a prime number p . a, b, and p are public information
G	G = (x1, y1) is a point on Elliptic curve with large order n s.t. nG = O G is publicly known.
N_a, N_b	private key of sender A, and receiver B respectively
P_a , P_b	Public Keys- for sender A, and receiver B respectively
K	Random integer K mutually agreed by Sender A, and Receiver B

Appendix B: List of Functions Used

Funcation Name	Description
function mSqrt(a, b, p, Xj)	Find a solution y to the Elliptic curve: y^2 = x^3 + a*x + b
function msgXvalue(Msg,k,a,b,p)	Encode a message to find x value on an Elliptic curve
function msgYvalue(Msg,k,a,b,p)	Encode a message to find y value on an Elliptic curve
function modInv(number, p)	Mod Inverse: Modular Inverse of number - using Mod p
function negativeG(y,p)	
function calculateLambda(a, x1, y1, x2, y2, p)	Find value of Lambda needed to calculate x and y values on Elliptic Curve for P + Q where P = (x1, y1) and Q = (x2,y2)
function xCordinate(a, x1, y1, x2, y2, p)	Find x value on Elliptic Curve for P + Q where P = (x1, y1) and Q = (x2,y2)
function yCordinate(a, x1, y1, x2, y2, p)	Find y value on Elliptic Curve for P + Q where P=(x1,y1) and Q = (x2,y2)
function nTimesGx(a,x1,y1,p,num)	scalar multiplication of G(x,y)
function nTimesGy(a,x1,y1,p,num)	scalar multiplication of G(x,y)
function decodingMessage (x,k)	Convert ciphertext to plaintext

function computeCiphertext (msg,pbX,pbY,gX,gY,K,a,b,p)	Given plaintext msg and senders Public key Pb, calculate Ciphertext
	// Also need G - Generator polynomial
	// Elliptic curve coefficients a,b
	// prime number p
	// random number K

function calculatePlaintext(a,p,Nb,K,xKg,yKg,cipherX, cipherY)	Calculate plaintext message