Cryptography and Network Security

Anirudh A C, Sanskar Deopuje, Nandhith Karthikeyan, Anandan Babu
Group-10

Mechanical Department, Amrita Vishwa Vidyapeetham Bengaluru
Kasavanahalli, Carmelaram P.O, Bengaluru-560035, India

bl.en.u4rae23003@bl.students.amrita.edu
bl.en.u4rae23010@bl.students.amrita.edu
bl.en.u4rae23027@bl.students.amrita.edu
bl.en.u4rae23055@bl.students.amrita.edu

Abstract – ***This paper delves into Cryptography, its history, and its adaptation to the modern world. It continues on about network security and discusses a few famous algorithms used in network security. It also compares algorithms used in the past and present. Furthermore, it explains how ECC can be used in drones, especially in the military, to strengthen security.***

I. Introduction

This paper deals with the applications and working cryptographic and network security algorithms. The protection of data has been one of the most prominent tasks since ages. Ciphers is a way of writing code in a secretive way. These ciphers were massively used in World War II. This paper focuses on various types of ciphers including Ceaser cipher, Monoalphabetic, Polyalphabetic, Playfair and Hill. These ciphers focus on encryption and decryption of data. Encryption and decryption are two means of protecting and decoding a set of data. The journey of ciphers began with the Ceaser cipher which was developed for military communications. Monoalphabetic cipher, which replaced each letter of the alphabet with another, and then the Playfair cipher, which utilized a 5*5 grid of letters for encryption. As ciphers advanced, ciphers like Hill cipher and Polyalphabetic cipher came to existence. With the advent of technology, cryptography also underwent many changes, a major evolution in cryptography is considered as AES, which is a symmetric encryption algorithm. Currently, asymmetric encryption algorithms, such as RSA and Diffie-Hellman are being utilised more, which use public key cryptography. Moreover, Elliptic Curve Cryptography (ECC) emerged as a powerful alternative for the older algorithms, offering smaller key sizes with same amount of security.

This paper provides a general overview on traditional and modern means of cryptography highlighting the importance of traditional methods and relating them with moderns means of network security.

II. Cryptography

In today's world, especially with the advent of technology and AI, data protection is vital, and its negligence can lead to a barrage of problems.

Cryptography is the act of protecting data by encoding it with the help of a certain key and cipher techniques to prevent leaks. The process of encoding data is called encryption. Encrypted data can be decrypted (the process of decoding) using the same/another key.

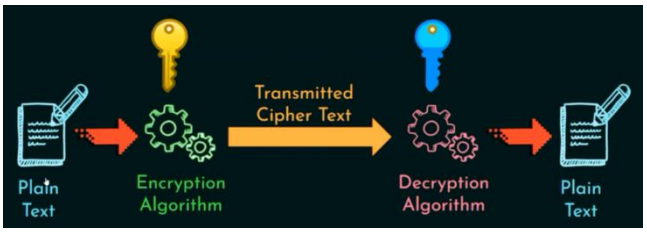


Fig. 1 Flowchart showcasing the method of [2]

“The art or science encompassing the principles and methods of transforming an intelligible message into one that is unintelligible and then retransforming that message back into its original form.”

Modern algorithms can involve complex mathematics like ‘Elliptical Curves’. The ciphers showcased in the first few sections can be easily broken with the help of modern computers.

1) *Symmetric Cryptography:* It uses the same key for both encryption and decryption. It is also called Private Key Cryptography.

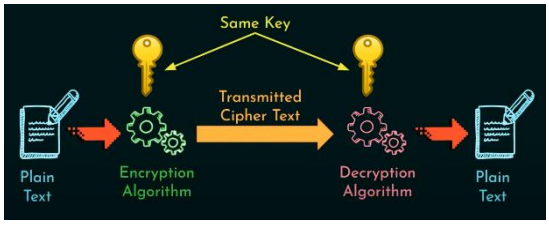


Fig. 2 Flowchart showcasing Symmetric Cryptography [2]

2) *Asymmetric Cryptography:* It uses different keys for encryption and decryption. It is also called Public Key Cryptography.

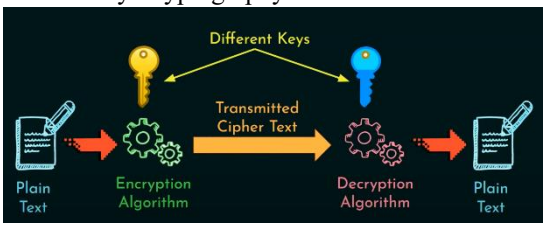


Fig. 3 Flowchart showcasing Asymmetric Cryptography [2]

Encryption schemes are of two types –

1) *Unconditionally Secure:* The cipher text does not contain enough information to determine the corresponding plane text uniquely.
2) *Computationally Secure:* The cost of breaking the cipher exceeds the value of the cipher text, and the time required by the attacker to break the cipher text exceeds the useful lifetime of the information.

A few common methods

- Cryptanalysis
- Brute-force attacks

A. Cryptanalysis

Cryptanalytic attacks are based on the information known to the cryptanalysis (or attackers). The cryptanalyst will focus mainly on finding the key used to generate the ciphertext.

Cryptanalytic attacks can fall under the following categories–

1) *Ciphertext only:* The cryptanalyst has access to the encryption algorithm and the ciphertext.
2) *Known Plaintext:* The cryptanalyst has access to the encryption algorithm, the ciphertext, and one or more plaintext-ciphertext pairs formed with the secret key.
3) *Chosen Plaintext:* The cryptanalyst has access to the encryption algorithm, the ciphertext, and the plaintext chosen by the cryptanalyst together with its ciphertext generated with the secret key.
4) *Chosen Ciphertext:* The cryptanalyst has access to the encryption algorithm, the ciphertext, and the ciphertext chosen by the cryptanalyst together with its decrypted plaintext generated with the secret key.
5) *Chosen Text:* The cryptanalyst has access to the chosen plaintext and chosen ciphertext.

B. Brute-force Attacks

Brute-force attacks consist of the attacker trying out every possible combination (key) to decrypt the data. Thus, the name 'Brute-force'. It consumes a lot of time and resources. It involves guessing and exhaustive key search.

There are many software tools available on the internet like-

- Aircrack-ng
- John the Ripper
- Rainbowcrack
- Ophcracck
- DaveGrohl
- Hydra
- Hashcat

And many more…

Do not use these softwares to harm anyone and use them only for study purposes.

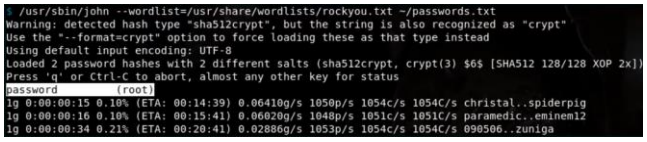


Fig. 4 Screenshot of John the Ripper tool performing brute-force attacks

Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) is a powerful invention that is used to prevent Brute-force attacks. It does this with the help of the following parameters-

- Cursor Tracking
- User History
- IP Address, etc.

Fig. 5 different types of CAPTCHA

III. Classical Encryption Techniques

Classical encryption techniques are methods used in history to protect the transfer of information. They are obsolete in the current world. They are vulnerable to attacks and easily decrypt ciphers.

Classical encryption techniques fall under two categories–

1) *Substitution:* Each plain text is substituted by other letters. For example,
   Plain Text – bag
   Cipher Text – XMA
   A few substitution techniques are –
   - Caeser Cipher
   - Monoalphabetic Cipher
   - Playfair Cipher
   - Hill Cipher
   - Polyalphabetic Cipher
   - One-time Pad
2) *Transposition:* Some sort of permutation is applied on the plain text letters. For examples,
   Plain Text – *mathematics*
   Cipher Text – IAHTEMSMTAC
   A few transposition techniques are –
   - Rail fence
   - Row column transposition

A. Caeser Cipher

Letters are replaced by other letters or symbols. This is one of the earlier known and the simplest method; it was used by Julius Caeser. In Caeser cipher, each letters is replaced by the letter three places down. For example,

A (00) → D (04)

B (02) → E (05)

Y (25) → B (02)

1) *Encryption:* For each plain letter 'p', the ciphertext letter 'C' is substituted
   $C = E(p, k) \bmod 26 = (p + k) \bmod 26$
   Plaintext – weekendz
   Key – 4
   Ciphertext – AIIOIRHD

2) Decryption: For each cipher text 'C', the plain letter 'p' is received
   $p = D(C, k) \bmod 26 = (C - k) \bmod 26$
   Ciphertext – RERCPQV
   Key – 17
   Plaintext – analyze

3) *Pros:* It is simple and easy to implement.
4) *Cons:* The encryption and decryption algorithms are widely known. There are only 25 keys to try which makes it vulnerable to brute force attacks. Furthermore, the language of the plaintext is known and is easily recognisable.

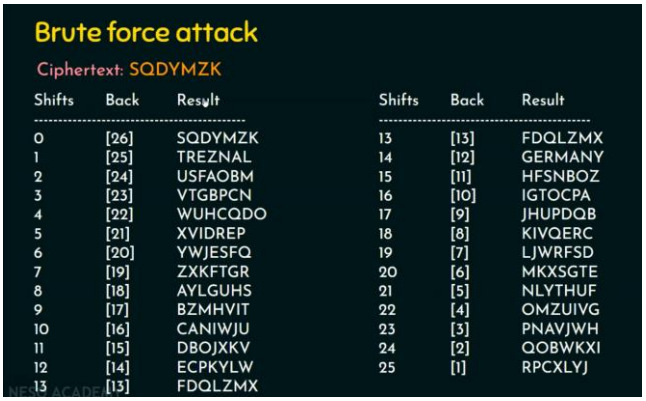


Fig. 6 Caeser Cipher is vulnerable to Brute Force attacks

B. Monoalphabetic Cipher

Monoalphabetic is a type of encryption technique in cryptography where each character of the plain text is mapped to another fixed character of the cipher text.

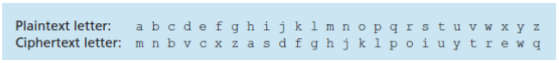
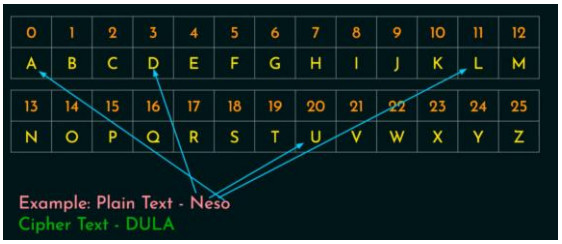


Fig. 7 example of plain text and there ciphertext

The 'cipher' line can be any permutation of the 26 alphabet characters. A single cipher alphabet (mapping from plain alphabet to cipher alphabet) is used per message.

Let's take an example:



Here the word is NESO, and in any random way each alphabet can be assigned to any other alphabet here.

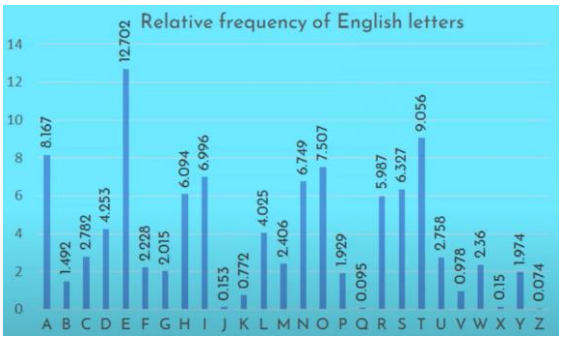
N→D

e →U

s →L

o → A

1) *Drawback:* letter frequency is the number of times of the alphabet appear on average in written language and given below is the letter frequency in English language.



Here we can see at average the alphabet 'E' appears the most and the letter 'X' appears the least in a sentence.

This makes easy to decrypt monoalphabetic cipher using the given data. Let's understand with an example:

GZGEWVGRNCP

Here we are taking a cipher, now it's clearly seen that 'G' appears the most in this cipher so the chances of 'G' being 'E' is high since 'E' has the highest frequency so, we put 'E' wherever 'G' exist

G	Z	G	E	W	V	G	R	N	C	P
E		E				E				

Now, since every other letter appears only once the next few letters can be attained though trial and error or guessing strategies

G	Z	G	E	W	V	G	R	N	C	P
E		E				E				
E		E			T	E				
E		E			T	E			A	
E		E			T	E		L	A	N
E		E			T	E	P	L	A	N
E	X	E	C	U	T	E	P	L	A	N

Here our plain text was, 'EXECUTEPLAN'.

2) *Pros:* Better security than Caesar cipher

3) *Cons:* monoalphabetic ciphers are easy to break because they reflect the frequency data of the original prone to guessing strategy using English letter frequency of occurrence of the letter

C. Playfair Cipher

The Playfair cipher, Playfair square, or Wheatstone-Playfair cipher, is a manual symmetric encryption technique and was the first literal diagram substitution cipher. The scheme was invented in 1854 by Charles Wheatstone but bears the name of lord Playfair for promoting its use.

The technique encrypts pairs of letters, instead of single letters as in the simple substitution cipher. The Playfair is thus significantly harder to break since the frequency analysis does not work with it.

1) *Constraints* The Playfair cipher uses a 5x5 matrix of letters (the key table), which contains no duplicates. The letters I and J are treated as the same letter
2) *Working:* we form the key table by placing the unique letters of a keyword in order, followed by the remaining letters of the alphabet.

Let us understand with an example by taking the word SECURITY as an example. First, we write down the letter of this word in the first square of the matrix,

S	E	C	U	R
I	T	Y		

We fill up the remaining squares of the matrix with the remaining letters of the alphabet and we treat I and J as one, since there are 26 letters, and we only have 25 squares available

S	E	C	U	R
I	T	Y	A	B
D	F	G	H	K
L	M	N	O	P
Q	V	W	X	z

While selecting a keyword we need to ensure that no letter is repeated and we also need to ensure that I and J don't appear together for example: INJUSTICE

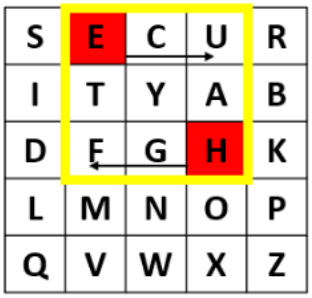
3) *Preparing the message:* when preparing the message 'HELLO WORLD', we get 'HELLOWORLD' notice we remove any non-alphabetic character, such as spaces or punctuation marks.
4) *Paring the letter:* if it contains identical consecutive letters, we insert X between them, and since the length of the string is odd we append another X at the end, HE LX LO WO RL DX

There are three rules for encrypting letters in the same pair

- If both letters in the pair are in the same column of the key square, we replace each letter with the letter to its right
- If both letter in the pair are in the same column of the key square, we replace each letter with the letter below it
- If the letters are in different rows and columns, we form a rectangle with the pair and replace each letter with the letter rectangle opposite corner.

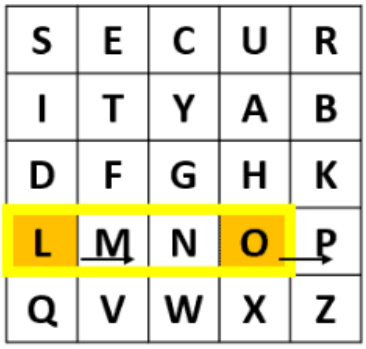
Let's understand these rules, by solving further our paired letters is HE LX LO WO RL DX

Let's first, do for HE:

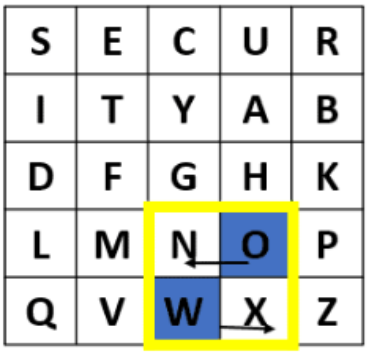


Now for LX:

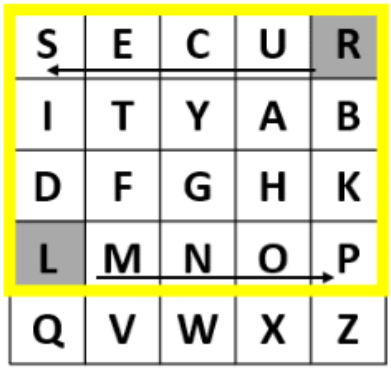
For LO:



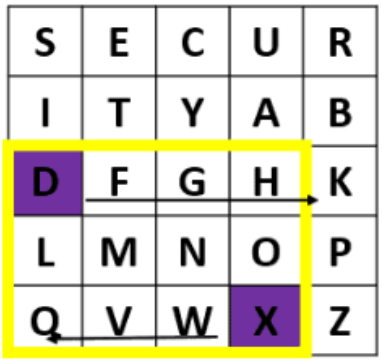
For WO:



For RL:



For DX:



After applying the encryption rules to all the letter pairs, we get FUOQMPXNSPHQ.

When it comes to the decryption a message encrypted with the Playfair cipher, the process involves reversing the entire thing applied during encryption.

5) Drawback: it relies on a 5x5 matrix of letters, making it unable to encrypt numbers, symbols, or non-alphabetic character

D. Hill Cipher

Hill Cipher is a multi-letter encryption process that provides better security as compared to the previous techniques. It was developed by American Mathematician and Cryptographer, Lester Hill, in the year 1929.

Hill Cipher can encrypt multiple letters at a time – digraph, trigraph, or polygraph.

1) *Modular Arithmetic:* It is an arithmetic system for integers involving remainders and modulus. For example,
   $10 \equiv 1 \pmod 3$ $\quad 10 + 16 \equiv 1 + 1 \pmod 3$
   $17 \equiv 1 \pmod 2$ $\quad 7 \times 7 \equiv 2 \times 2 \pmod 5$

2) *Encryption:* For each cipher text 'C', the plain letter 'p' is received

$$C = E(k, p) = p \times k \pmod{26}$$

$$(C_1\ C_2\ C_3) = (P_1\ P_2\ P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \bmod 26 \quad \text{Encryption}$$

$$(c_1 = k_1 p_1 + k_4 p_2 + k_7 p_2 \quad c_2 = k_2 p_1 + k_5 p_2 + k_8 p_2 \quad c_3 = k_3 p_1 + k_6 p_2 + k_9 p_2)$$

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} = \begin{bmatrix} 67 \\ 222 \\ 319 \end{bmatrix} \equiv \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} \pmod{26}$$

Plaintext – ACT
Key – GYBNQKURP
Ciphertext – POH

3) *Decryption:* For each cipher text 'C', the plain letter 'p' is received

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix}^{-1} \equiv \begin{bmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{bmatrix} \pmod{26}$$

$$k^{-1} = (\text{Det } k)^{-1} \times \text{Adj } k$$

$$p = D(k, C) = C\ k^{-1} \pmod{26} = p \times k \times k^{-1} \pmod{26}$$

Ciphertext – POH
$(Key)^{-1}$ – IFKVIVVMI
Plaintext – ACT

4) *Pros:* High levels of security and efficiency. It is simple

$$\begin{bmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{bmatrix} \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} \equiv \begin{bmatrix} 260 \\ 574 \\ 539 \end{bmatrix} \equiv \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} \pmod{26}$$

and conceals letter frequency making it more robust. It is strong against ciphertext attacks.

5) *Cons:* It is vulnerable and easy to solve and is weak against modern cryptography. Keys must be invertible and block sizes are smaller.

In Hill Cipher, computers can encrypt and decrypt data simultaneously with the help of Linear Algebra

E. Polyalphabetic Cipher

A polyalphabetic cipher is a substitution, using multiple substitution alphabets. The Vigenère cipher is probably the best-known example of a polyalphabetic cipher, through it is a simplified special case. The Enigma machine is a more complex but is still fundamentally a polyalphabetic cipher.

Polyalphabetic cipher improves on the simple monoalphabetic technique that we saw before

1) Common features: A set of related monoalphabetic substitution rules is used. A key determines which rule is chosen for a given transformation.

2) Vigenère cipher: it consists of the 26 Caesar ciphers with shifts of 0 through 25.

Encryption process: $C_i = (P_i + K_{i \bmod m}) \bmod 26$

Decryption process: $P_i = (C_i - K_{i \bmod m}) \bmod 26$

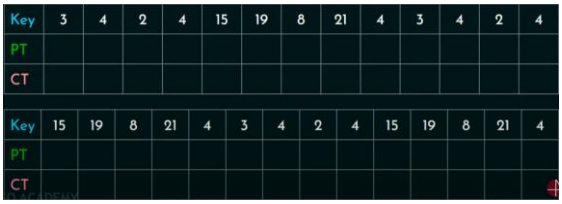
Where, P is plain text
C is cipher text
K is key

Let us understand with an example:

Key : deceptivedeceptivedeceptive
Plaintext : wearediscoveredsaveyourself

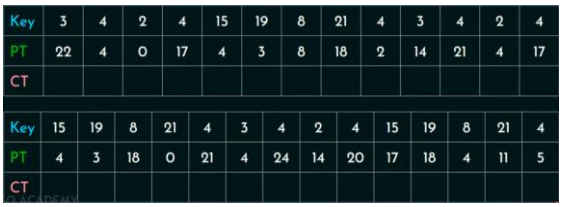
We will take the above key for the respective plaintext, now let's make cipher text. To do that let's recall which alphabet used to hold which integer value in Caesar cipher i.e. A→0

B→1
C→2
and so on...



Key	3	4	2	4	15	19	8	21	4	3	4	2	4
PT													
CT													
Key	15	19	8	21	4	3	4	2	4	15	19	8	21	4
PT														
CT														

so, for all alphabet we have an equivalent digit and they are filled respectively, we will repeat the same process for plain text



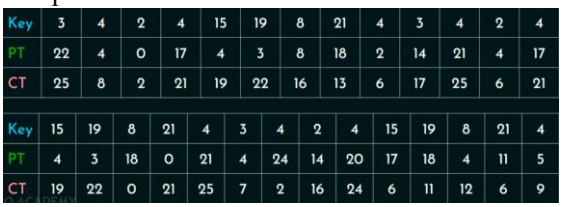
Key	3	4	2	4	15	19	8	21	4	3	4	2	4
PT	22	4	0	17	4	3	8	18	2	14	21	4	17
CT													
Key	15	19	8	21	4	3	4	2	4	15	19	8	21	4
PT	4	3	18	0	21	4	24	14	20	17	18	4	11	5
CT														

Now let's use the formula for encryption, for the first alphabet i.e. C1

C1= (P1 + K1) mod26
C1= (22+3) mod26
C1= 25

Similarly, lets solve for C8

C8= (21 +18) mod26
C8= (39) mod26
C8= 13
Continue the same process for others and we get the below as cipher text.



Key	3	4	2	4	15	19	8	21	4	3	4	2	4
PT	22	4	0	17	4	3	8	18	2	14	21	4	17
CT	25	8	2	21	19	22	16	13	6	17	25	6	21
Key	15	19	8	21	4	3	4	2	4	15	19	8	21	4
PT	4	3	18	0	21	4	24	14	20	17	18	4	11	5
CT	19	22	0	21	25	7	2	16	24	6	11	12	6	9

IV. Network Security

In our increasingly digitized world, where information is exchanged at unprecedented rates, the significance of cryptography cannot be overstated. Cryptography, the art and science of secure communication, has long been a cornerstone of civilization's efforts to protect sensitive information. From ancient times, where ciphers were used to encode military strategies, to the modern era, where cryptographic algorithms safeguard financial transactions, personal data, and national security, cryptography has played a pivotal role in shaping history and ensuring privacy. Its relevance extends to diverse fields including finance, communication, and cybersecurity. At its core, cryptography relies heavily on principles of linear algebra,

1) *Confusion:* Confusion refers to the process of making the relationship between the plaintext and the ciphertext as complex and obscure as possible. In cryptographic terms, this means that even a small change in the plaintext input should result in significant changes throughout the ciphertext output.
2) *Diffusion:* Diffusion involves dispersing the influence of individual plaintext characters or bits throughout the entirety of the ciphertext. In other words, diffusion ensures that the effects of each plaintext character or bit are spread out across the entire ciphertext, making it difficult for an attacker to discern patterns or extract meaningful information. Diffusion increases the complexity and randomness of the ciphertext, thereby making it more resistant to cryptanalysis techniques.
3) *Plaintext:* Plaintext is the original, unencrypted data or message that is readable and understandable by humans or machines
4) *Ciphertext:* Ciphertext is the encrypted form of plaintext, obtained by applying cryptographic algorithms and keys. It appears as scrambled or unintelligible data, making it unreadable to anyone without the proper decryption key.
5) *Encryption:* Process that transforms plaintext into ciphertext, rendering it secure for transmission or storage.

A. Advanced Encryption System

1) *Key Expansion:* AES encrypts messages in blocks of 128 bits.
   It allows 3 different key lengths i.e. 128, 192, 256.
   The number of rounds in Encryption and Decryption depends on the key length.
   128 bit: 10 rounds
   192 bit: 12 rounds
   256 bit: 14 rounds

   In our example, we'll be taking the key length as 128.

   Let us take the key as Geetha Mam Rocks

   We first take each character of the string that is and convert each letter to binary (including the space)

   Then we take the binary number and divide it into 2 parts and convert each part to a hexadecimal value for

example G will become 01000111 and be split into (0100,0111) which will be converted into (4,7)

G	e	e	t	h	a		M
010 001 11	011 001 01	011 001 01	011 101 00	011 010 00	011 000 01	001 000 00	010 011 01
47	65	65	74	68	61	20	4D
b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8

a	m		R	o	c	k	s
011 000 01	011 011 01	001 000 00	010 100 10	011 011 11	011 000 11	011 010 11	011 100 11
61	6D	20	52	6F	63	6B	73
b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}	b_{16}

Now we will make this into a 4 x 4 matrix where the elements are filled column-wise.

$$\begin{bmatrix} b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \\ b_4 & b_8 & b_{12} & b_{16} \end{bmatrix} = \begin{bmatrix} 47 & 68 & 61 & 6F \\ 65 & 61 & 6D & 63 \\ 65 & 20 & 20 & 6B \\ 74 & 4D & 52 & 73 \end{bmatrix}$$

In this matrix, each column is known as a 'word', where the first column is w_0 and second is w_1 and so on.

As we have previously discussed, key expansions take 10 steps for a 128-bit key.

So we need to have w_0 to w_{43} where the first 4 words i.e. w_0, w_1, w_2, w_3 is the initial key or round 0, and the consecutive group of 4 words makes the next round, and each group is known as a subkey.

Now, how do we get the 10 subkeys?

The first subkey (round 1) consists of w_4, w_5, w_6, w_7 .

To get w_4 we pass the value of w_3 to a function g , take w_0 perform the XOR operation on both of them.

To get w_5 we perform XOR with w_1 & w_4 and to get w_6 we perform XOR with w_2 & w_5 and so on.

Function g:

Function g takes a column $\begin{bmatrix} 6F \\ 63 \\ 6B \\ 73 \end{bmatrix}$ and rearranges it as $\begin{bmatrix} 73 \\ 6F \\ 63 \\ 6B \end{bmatrix}$ by moving each element to the next row and the 4th element to the first row.

Then using table called the S-box, it converts each element into its corresponding value from the Rijndael S-box.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
10	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
20	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
30	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
40	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
50	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
60	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
70	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
80	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
90	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A0	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B0	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C0	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D0	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E0	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F0	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Fig. n S-Box

So $\begin{bmatrix} 6F \\ 63 \\ 6B \\ 73 \end{bmatrix}$ this becomes $\begin{bmatrix} A9 \\ FB \\ 7F \\ 8F \end{bmatrix}$

XOR stands for "exclusive OR". It's a logic operation that compares two binary digits.

For the next step, we use a table for each round, where each column is a round constant:

- If both digits are the same, the result is 0.
- If the digits are different, the result is 1.

R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
01	02	04	08	10	20	40	80	1B	36
00	00	00	00	00	00	00	001	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

$$\begin{bmatrix} 48 \\ FB \\ 7F \\ 8F \end{bmatrix} \text{XOR} \begin{bmatrix} 01 \\ 00 \\ 00 \\ 00 \end{bmatrix} \Rightarrow \begin{bmatrix} 10101000 \text{ XOR } 00000001 \\ 11111011 \text{ XOR } 00000000 \\ 01111111 \text{ XOR } 00000000 \\ 10001111 \text{ XOR } 00000000 \end{bmatrix} \Rightarrow \begin{bmatrix} 10101000 \\ 11111011 \\ 01110100 \\ 10000100 \end{bmatrix} = \begin{bmatrix} A9 \\ FB \\ 7F \\ 8F \end{bmatrix}$$

The final column matrix that we get is the function g(w_3).

Now, $w_4 = w_0$ XOR $g(w_3)$

$$w_5 = w_4 \text{ XOR } w_1$$

$$w_6 = w_5 \text{ XOR } w_2$$

$$w_7 = w_6 \text{ XOR } w_3$$

$$\begin{matrix} w_0 & w_1 & w_2 & w_3 \end{matrix} \quad \begin{matrix} w_4 & w_5 & w_6 & w_7 \end{matrix}$$
$$\begin{bmatrix} 47 & 68 & 61 & 6F \\ 65 & 61 & 6D & 63 \\ 65 & 20 & 20 & 6B \\ 74 & 4D & 52 & 73 \end{bmatrix} \begin{bmatrix} 28 & 40 & 21 & 4E \\ 06 & 67 & 0A & 69 \\ 0E & 2E & 0E & 65 \\ 07 & 4A & 18 & 6B \end{bmatrix} (subkey\ 1)$$

This is the completion of 1 round of key expansion.

Now suppose we want to convert a message of length 128 i.e MathematicsClass

We convert it to a hexadecimal matrix like we did for the key:

MathematicsClass becomes 4D 61 74 68 65 6D 61 74 69 63 73 43 6C 61 73 73

$$\begin{bmatrix} 4D & 65 & 69 & 6C \\ 61 & 6D & 63 & 61 \\ 74 & 61 & 73 & 73 \\ 68 & 74 & 43 & 73 \end{bmatrix}$$

Which we XOR with subkey 1which is

$$\begin{bmatrix} 28 & 40 & 21 & 4E \\ 06 & 67 & 0A & 69 \\ 0E & 2E & 0E & 65 \\ 07 & 4A & 18 & 6B \end{bmatrix}$$

The result that we get will is called a state key

$$\begin{bmatrix} 65 & 25 & 48 & 22 \\ 67 & 0A & 69 & 08 \\ 7A & 4F & 7D & 16 \\ 6F & 3E & 5B & 18 \end{bmatrix}$$

1) *Encoding:* State key that we get will then go through steps:
   - Byte substitution
   - Row Shift
   - Mix Columns
   - Add round key

Byte substitution: We use the S-Box again to convert each element of the Statekey.

$$\begin{bmatrix} 65 & 25 & 48 & 22 \\ 67 & 0A & 69 & 08 \\ 7A & 4F & 7D & 16 \\ 6F & 3E & 5B & 18 \end{bmatrix} = \begin{bmatrix} 4D & 3F & 52 & 93 \\ 85 & 67 & F9 & 30 \\ DA & 84 & FF & 47 \\ A8 & B2 & 39 & AD \end{bmatrix}$$

Row shift: We don't shift the first row shift every element in the 2nd row to the left by 1 unit, 3rd row by 2 units and 4th row by 3 units .

$$\begin{bmatrix} 4D & 3F & 52 & 93 \\ 85 & 67 & F9 & 30 \\ DA & 84 & FF & 47 \\ A8 & B2 & 39 & AD \end{bmatrix} \rightarrow \begin{bmatrix} 4D & 3F & 52 & 93 \\ 30 & 85 & 67 & F9 \\ FF & 47 & DA & 84 \\ B2 & 39 & AD & A8 \end{bmatrix}$$

Mix Columns: We then matrix multiply the row shifted matrix by another matrix.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} * \begin{bmatrix} 4D & 3F & 52 & 93 \\ 30 & 85 & 67 & F9 \\ FF & 47 & DA & 84 \\ B2 & 39 & AD & A8 \end{bmatrix} = \begin{bmatrix} r_1 & r_5 & r_9 & r_{13} \\ r_2 & r_6 & r_{10} & r_{14} \\ r_3 & r_7 & r_{11} & r_{15} \\ r_4 & r_8 & r_{12} & r_{16} \end{bmatrix}$$

The way we multiply this is assume each element as a binary number, for e.g. in this case,

02=00000010

42=01000010

We convert the binary number to a polynomial where each term corresponds to the coefficients of said polynomial.

$$02 = 00000010 = x$$

$$4D = 01001101 = x^6 + x^3 + x^2 + 1$$

Then we multiply both polynomials and get another polynomial which is $(x^7 + x^2)$ which is then converted into another binary number 10000100.

$$r_1 = (02 \times 4D) + (03 \times 30) + (01 \times FF) + (01 \times B2)$$

Like this, matrix multiplication is carried out to get $r_1, r_2 \ldots r_{16}$.

Add Round Key: In this step, we XOR the r matrix with the subkey we got for the first round in key expansion and we do this again 9 times to get the ciphertext. Note that in the last round, we do not mix columns.

2) *Decoding:* While decoding, the cipher text will then go through the following steps:
   - Inverse S-Box Substitution
   - Row Shift
   - Mix columns
   - Add Round key

Inverse Byte substitution: This time, we use an Inverse S-Box again to convert each element back into the statekey

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
70	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
A0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
B0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
C0	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
D0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
E0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
F0	**8C**	**A1**	**89**	**0D**	**BF**	**E6**	**42**	**68**	**41**	**99**	**2D**	**0F**	**B0**	**54**	**BB**	**16**

Row shift: During the encoding process, we shifted rows to the left, now we do the opposite by shifting rows to the left.

We don't shift the first row.

We shift every element in the 2nd row to the right by 1 unit,3rd row by 2 units and 4th row by 3 units

Mix Columns: We then matrix multiply the matrix by another matrix similar to the time we mixed column in encoding.

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} * \begin{bmatrix} r_1 & r_5 & r_9 & r_{13} \\ r_2 & r_6 & r_{10} & r_{14} \\ r_3 & r_7 & r_{11} & r_{15} \\ r_4 & r_8 & r_{12} & r_{16} \end{bmatrix}$$

Add Round Key: In this step, we XOR the r matrix with the subkey we got for the first round in key expansion.

We do this again 9 times to get the plaintext back from the ciphertext.

B. RSA

RSA is an asymmetric cryptographic algorithm, this means that it works on 2 different keys namely;

- Public Key
- Private key

The public key is known to everyone and the private key is kept private

The fundamental principle by which the RSA works is the fact that it is difficult to factorize a large number.

The public key is the product of 2 large prime numbers and the private key is also derived from the 2 prime numbers. The strength of encryption completely depends on the size of the key.

Generating the Public Key: Let us select 2 prime numbers $P = 53$ and $Q = 59$. Now to make the Public key, we need 2 variables n, e where $n = P * Q$ and $1 < e < \phi(n)$ and e must not be a factor $\phi(n)$.

$$\phi(n) = (P-1)(Q-1) = 3016$$

Let us take e as 3.

Generating the Private key: The private key is denoted by d and is equal to

$$d = \frac{x * \phi(n) + 1}{e}$$

x here is a random integer. Let's suppose $x = 2$, then the value of $d = 2011$.

Encryption: Suppose we want to encrypt the text "HI" let us consider $\mathrm{H} = 8,\ \mathrm{I} = 9$.

$$Ciphertext\ (c) = (89^e) mod\ n = 1349$$

Decryption: To decrypt this data, we use the following formula:

$$Plaintext(p) = (c^d) mod\ n = 89$$

Proof of RSA

Encryption: $p^e\ mod\ n = c$

Decryption: $c^d mod\ n = p$

Where M is less than N and coprime to it. So we must prove that $(\ p^e\ mod\ n)^d mod\ n = p$

To prove RSA, we must be aware of 2 concepts

- Modular Arithmetic Exponentiation law
- Euler's Law

Modular Arithmetic Exponentiation Law: When $A\ , k\ , n$ are positive integers where $A < n$ then:

$$A^k mod\ n = (A\ mod\ n)^k mod\ n$$

This is true because, if $A < n$ then $A\ mod\ n = A$, because if you divide A by n, since n is larger than A, you divide zero times, and the remainder is A itself.

Euler's Law: $(e * d) = \big(\phi(n)\big) * x + 1$

"When A and n are coprime, positive integers, $\phi(n)$ is termed a "*Euler's totient"* and x is a positive integer, then $A^{\phi(n)x}\ mod\ n = 1$ "

Now in $A^k mod\ n = (A\ mod\ n)^k mod\ n$

let $A = p^e, k = d$

then $(p^e mod\ n)^d mod\ n = p^{ed} mod\ n$ (MAEL)

Now

$$p^{ed} \; mod \; n \rightarrow p^{(\phi(n)*x+1)} \; mod \; n \rightarrow p * p^{\phi(n)*x} * mod \; n \rightarrow p$$

Here we have proved that $(p^e mod \; n)^d mod \; n = p$.

V. Elliptic Curve Cryptography

ECC, as the name implies, is an asymmetric encryption algorithm that employs the algebraic architecture of elliptic curves with finite fields.

- Elliptic Curve Cryptography (ECC) is an encryption technology comparable to RSA that enables public-key encryption.
- ECC uses the mathematical concept of elliptic curves for the safety and security of data.
- ECC provides a better alternative for keys allowing them to get stored in less space comparatively.

A. History

Elliptic curve cryptography was introduced in 1985 by Victor Miller and Neal Koblitz who both independently developed the idea of using elliptic curves as the basis of a group for the discrete logarithm problem. Up until 1970's, all the encryption in use around the world was based upon the symmetric techniques of cryptography in which only a single key is used altogether. During the second world war when online communication was a source of information, asymmetric cryptographic methods or public key cryptography came into existence with first being the Diffie Hellman. Further in 1985 ECC was developed based on asymmetric means which introduced a new degree of security to public key cryptosystems, that provide combined encryption and digital signature services.

ECC algorithms entered wide use from 2004 to 2005. Elliptic curves found their use as these curves are almost impossible to decrypt given the infinite number of combinations which are possible, which make them almost impossible to decrypt. Due to this property elliptic curve cryptography found its usage in many sectors like bitcoin technology, banking systems, etc. Elliptic curve cryptography thus now is one of the most widely used cryptographies, given the inability of RSA to provide total safety and security.

B. Types of Encryptions

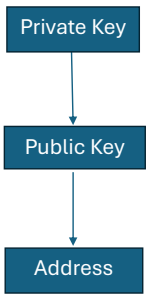
1) *Symmetric Key Encryption:* In symmetric key encryption the message is encrypted using a key and the same key is used to decrypt the message which makes it easy to use but less secure. The length of the key used is 128 or 256 bits. The Mathematical Representation is P = D (K, E (K, P)), where K is the encryption and decryption key

P is plain text
D is Decryption
E (K, P) is the Encryption of plain text using K.

2) *Asymmetric Key Encryption:* Asymmetric key encryption uses two keys namely public key and private key to encrypt and decrypt the message which adds up to it being more secure. The length of the key used is 2048 or higher but provides better resource utilization. The Mathematical Representation is

P = D (Kd, E (Ke,P)), where Ke is the encryption key
Kd is the decryption key
D is Decryption
E(Ke, P) is Encryption of plain text using encryption key
Ke. P is plain text

Elliptic Curve Cryptography uses Asymmetric Cryptography Technique to encrypt and decrypt the messages. In this method, the private key is used to derive the public key on the basis of which the messages are then encrypted and decrypted.



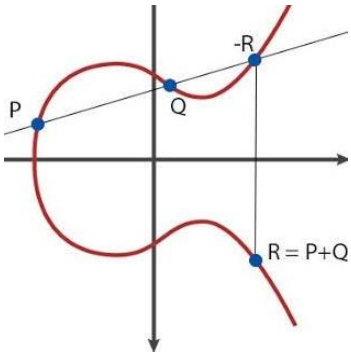
When a message is traded, a private key is generated first. From the private key the public key is generated. The public key then generates the address of the message. One cannot generate the private key from the address of the message because then it would decrease the security of the process.

C. Working of ECC

In mathematics, an elliptic curve is a smooth, projective, algebraic curve of genus one, on which there is a specified point O. An elliptic curve is defined over a field K and

describes points in K2, the Cartesian product of K with itself. Elliptic curve E(F) is a set of points in a filed F, satisfying an equation of the form: y.^2 + a1xy + a2y = x.^3 + a3x.^2 (3.1) + a4x + a5. This equation can be simplified further if the characteristic of the field is also different than 3, thus giving the more familiar equation, known as the Weierstrass normal form:

y.^2 = x.^3 + ax + b where a and b are constants.



Properties and Basics

- Elliptic curves are symmetric around X axis.
- A line through the Elliptic curve intersects it at a maximum three points.
- A mathematical function called as 'dot' is used in the elliptic curve cryptography.
- A line called as 'max' is used to define the key size of the curve.

Consider two points P and Q on the given elliptic curve. One more point -R is taken as the point of intersection of the line passing through P and Q and the curve.

The reflection of point -R along the X axis will be R. R will be equal to the sum of P and Q, i.e., R=P+Q. This relationship between the points P, Q and R is given by 'dot' which states that

P 'dot' Q =R.

In Elliptic Curve Cryptography same point consider P is dotted 'n' times with itself.

Max- Max is the line which is used to define the maximum extent to which the point should be dotted in x-direction in order to obtain the curve and encrypt the message. As the max increases, it becomes more difficult to crack the key as the number of solutions possible for a particular varies a lot. Starting from point P, it is dotted with itself for 'n' number of times until the max is reached.

1) *Private Key:* The point P is dotted for a secret number of times, i.e., 'n'. In ECC 'n' is considered as the private key which is impossible to find given the number of solutions and combinations possible.
2) *Public Key:* The function sketching out the elliptic curve, the start point, and the end point are the knowns in ECC which together are considered as the public key.

For example, consider the following curve for the elaboration of elliptic curve cryptography.

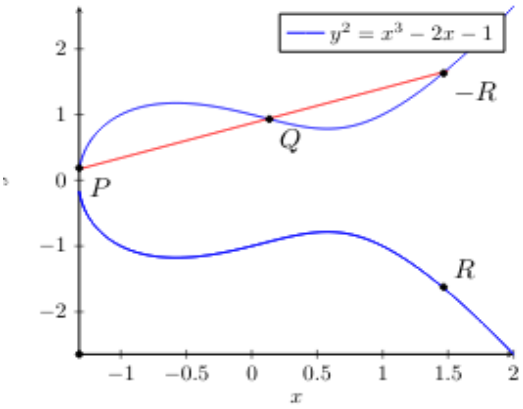


Fig. n Elliptic Curve in R2

Given points P and Q on the curve E, we draw a line between P and Q. If P(x1, y1) and Q(x2, y2) are distinct points of E, then we can express the coordinates of the point R(x3, y3) by solving for the intersection between the line going through P and Q and the curve E:

$$y = (y_2 - y_1) \cdot x + y_1 x_2 - \frac{y_2 x_1}{x_2 - x_1}$$

$$y^2 = x^3 + ax + b$$

Then the coordinates of R $(x_3, -y_3)$ are given by:

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_2 - x_1$$

$$y_3 = -\left(y_2 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right) * (x_3 - x_2)\right)$$

If P = Q, then we take the intersection of the tangent line and the curve E, which gives a simpler expression for R(x2, y2):

$$x_2 = \frac{3x_1^2 + a}{2y_1} - 2x_1$$

$$y_2 = -\left(y_1 + \left(\frac{3x_1^2 + a}{2y_1}\right) * (x_2 - x_1)\right)$$

Note: This expression includes the parameter a of the curve and not b as we take the derivative of the curve to get the slope of the tangent line, and b vanishes. Using this operation, we define scalar multiplication on the group G, by repeatedly adding a point to itself: nP = P + P + ... + P . One can compute nP efficiently using a variation of repeated squaring, the double and add method.

D. ECC Algorithms: Digital Signature Algorithms - Elliptic Curve Digital Signature Algorithm (ECDSA)

The Elliptic Curve Digital Signature Algorithm (ECDSA) is a signature algorithm based on Elliptic Curve Cryptography (ECC). It is the cryptographic algorithm used to create keys and create and verify signatures used for authentication.

The elliptic curve used in ECDSA has:

• Generator point: G is a specific point on the elliptic curve: (x = 55066263022277343669578718895168534326250603453777594175500187360389116729240, y=32670510020758816978083085130507043184471273380659243275938904335757337482424).

• Order: Order n of the subgroup of elliptic curve points, generated by G, which defines the length of the private keys (e.g. 256 bits) and is a prime number: 1157920892373161954235709850086879078528375642790749043826051631415181614943 37.

1) *Signatures:* The signature can be notated as (r, s, v):

   • r (integer in the range [1...n-1]): represents the x-coordinate on the elliptic curve based on a random point R on the curve.

   • s (integer in the range [1...n-1]): serves as proof of the signer's knowledge of the private key p. s is calculated using a random unique nonce k, ensuring that the signature is unique every time.

   • The value v is used to recover public keys from the value of r and represents the index of the point on the elliptic curve used for the signature. The recovery process generates multiple solutions for the possible value of R. The addition of v specifies which solution is required.

ECDSA is used for the following:

- Key generation
- Signing messages
- Signature verification

2) *Key Generation:* An ECDSA key pair consists of the following:

   • A private key (integer) p: generated as a random integer in the range [0…n-1]

   • A public key (EC point): a point on the elliptic curve pubKey = p * G

The signature (r, s, v) secures the private key due to the complexity of the Elliptic Curve Discrete Logarithm Problem (ECDLP). This problem involves finding an integer p (the private key) such that

pG = Q, where G is the Generator point and Q is the public key.

Given the public key and the generator point, it is computationally infeasible to derive the private key, therefore providing security for the ECDSA. One way to understand this is by taking two prime numbers and multiplying them together e.g.

134669 * 2467 = 332228423

Given the result, it would be very difficult (practically impossible) to calculate the input prime numbers, even for a computer.

3) *Signing Messages:* The ECDSA signing algorithm takes a message and a private key and produces a signature, a pair of integers (r, s).The ECDSA signing algorithm works by:
   - Calculating the message hash, using a cryptographic hash function e.g. SHA-256.
     h = hash(msg)
   - Generating securely a random number k.
   - Calculating the random point R = k * G and take its x-coordinate: r = R.x.
   - Calculating the signature proof s using the formula:
     s = k^-1 * (h + p * r) mod n
     where p is the signer's private key, and the order n
   - Return the signature (r, s).

4) *Signature Verification:* The ECDSA signature verification algorithm takes the signed message msg and the signature produced from the signing algorithm and the public key pubKey. The output is a boolean representing whether the signature is valid or invalid.The ECDSA signature verification algorithm works by converting s back to R (R') using the public key and message hash. The recovered R's x-coordinate r' is compared with r from the signature:

- Calculating the message hash, with the same hash function used when signing.
- Calculating the modular inverse s1 of the signature proof.

  s1 = s^-1 (mod n).
- Recovering the random point used during the signing.

  R' = (h * s1) * G + (r * s1) * pubKey.
- Retrieving r' from R': r' = R'.x.
- Calculating the signature validation result by comparing whether r' == r.

E. ECC Algorithms: Digital Signature Algorithms - Edwards-curve Digital Signature Algorithm (EdDSA)

The Edwards-curve Digital Signature Algorithm (EdDSA) was proposed as a replacement for the Elliptic Curve Digital Signature Algorithm for performing fast public-key digital signatures (ECDSA). Its primary benefits for embedded devices are higher performance and simple, secure implementations. During a signature, no branch or lookup operations based on the secret values are performed.

```
X: 0x9ed517414b512c920bb8f937cc1bca9f41d969b8343af6bec570aab0ab32a5871
Y: 0x843b2e0d08b8d27755435dd5317ea902ba0693d7edf1e8fff86103f043d648da1
Currently exchange the publickey (e.g. through Internet)
A shared key : 0x4e5e1ba418ace2566f0a2e5121a63a9631b2525cdf80147a601555caed0201f91
(B) shared key : 0x4e5e1ba418ace2566f0a2e5121a63a9631b2525cdf80147a601555caed0201f91
Equal shared keys: True
```

Fig. n Output of the python program for ECC

The above-mentioned Python code generates an ECC private-public key pair for the recipient of the message (based on the brainpoolP256r1 curve), then derives a secret shared key (for encryption) and an ephemeral cipher–text key (for ECDH) from the recipient's public key, and then derives same secret key pair (for decryption) from the recipient's secret key and the previously generated ephemeral ciphertext public key.

F. Applications of ECC

1) *Online Banking:* When we make an online purchase with your debit or credit card, your information is often encrypted using ECC before it's sent over the internet. This ensures that your information remains confidential and secure throughout the transaction process.
2) *Email Encryption:* Pretty Good Privacy (PGP) is a popular email encryption software that can leverage ECC to protect your emails from being read by anyone other than the intended recipient. PGP works by generating a public/private key pair for each user. The public key can be shared with anyone, but the private key must be kept confidential at all times. To encrypt an email, you simply need the recipient's public key; conversely, you'll need your private key to decrypt an email you've receive.
3) *Bitcoin:* A Bitcoin wallet uses ECC to generate a pair of keys: a private key, which is a randomly generated number, and a public key, derived from this private key using elliptic curve multiplication. The public key is your wallet address, shareable without compromising security.

   An elliptic curve equation is selected, and a curve is generated from it on a graph. This curve has unique mathematical properties that make it hard to crack. So, a starting point on the elliptic is chosen somewhere on the curve. This is the private key, which is used to encrypt messages.

   The public key is then generated from the private key and used to decrypt messages. These mathematical properties are what make elliptic curves secure.

VI. Diffie Hellman Cryptographic Algorithm

Diffie Hellman Algorithm is a key exchange protocol which allows the users to establish a shared secret that can be used for private communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

This algorithm helps two users to exchange keys among themselves. Diffie Hellman undertakes four variables for simplicity and practical applications of the algorithm.

1) *Public Key:* P and G are public keys in Diffie Hellman Algorithm which are both publicly available numbers.
2) *Private Key:* Users (say Alex and Ben) choose their own private numbers (a, b). These private numbers a and b are considered as private key for Alex and Ben respectively in Diffie Hellman Algorithm.

A. Explanation

Public Keys available for both Alex and Ben: P, G

Private Key:

Alex-a
Ben-b

Key generated:

Alex- x=G^amodP
Ben- y=G^bmodP

The exchange of keys that are generated takes place.

Key received:

Alex- y
Ben- x

Generated Secret Key:

Alex- ka=y^amodP
Ben- kb=x^bmodP

Algebriacally,
ka=kb
Both Alex and Ben would now have secret key known only to both of them and users would have a symmetric secret key to encrypt.

Consider an example for a better understanding of the Diffie Hellman Algorithm:

Alex and Ben have public keys: P=23, G=9.
Both of them selected private keys a=4 and b=3 respectively.

Alex and Ben compute public values:

Alex: x = (9^4 mod 23) = (6561 mod 23) = 6; Ben: y = (9^3 mod 23) = (729 mod 23) = 16

Alex and Ben exchange public numbers

Alex receives public key y =16 and Ben receives public key x = 6

Alex and Ben compute symmetric keys

Alex: ka = y^a mod p = 65536 mod 23 = 9
Ben: kb = x^b mod p = 216 mod 23 = 9
9 is the shared secret

B. Implementation

```
The value of P : 23
The value of G : 9
The private key a for Alice : 4
The private key b for Bob : 3
Secret key for the Alice is : 9
Secret key for the Bob is : 9
```

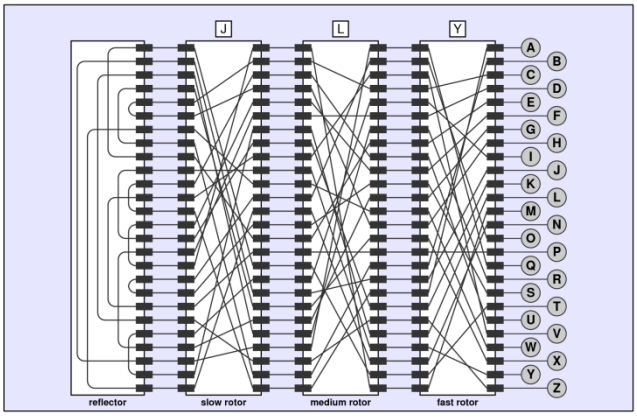
Fig. n Output of the C++ program for the Differ-Hellman Key Exchange algorithm

VII. CASE STUDY: THE ENIGMA MACHINE

A. Encryption

During World War II, a group of British Mathematicians and code-breakers who were working at a secret facility called Bletchley Park were attempting to break the German code for Navy communication, known as the Enigma.

Figure 3. Structural diagram of the Enigma machine showing the rotor setting JLY



The Enigma was a mechanical encryption device, used to encode and decode secret messages.

It was invented by Arthur Scherbius, who was a German engineer. It looked like a typewriter.

It consisted of a keyboard for inputting plaintext letters.

It had 3 rotors that scrambled the input by linking each plaintext letter to its corresponding ciphertext letter.

As a key was pressed, an electrical current would pass through the rotors, causing them to rotate, which would change the wiring configuration and therefore the encryption of subsequent letters. This rotation occurred with each key press, creating a constantly shifting encryption pattern.

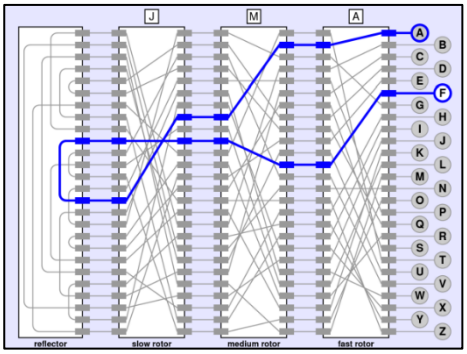


Fig. n The Enigma Machine after pressing A key once

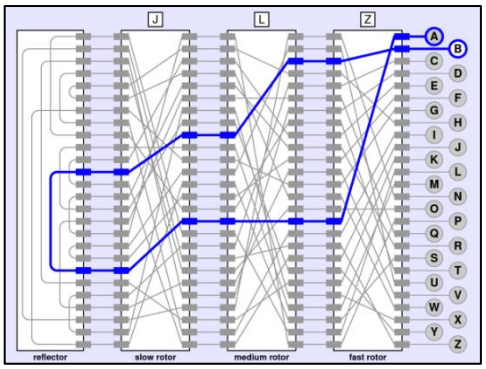


Fig. n The Enigma Machine after pressing A second time

The machine also included a plugboard, which increased the complexity of the encryption by creating 10 pairs of letters that could be swapped.

To decrypt the message, one needed to know the exact settings of the rotors and plugboard used by the sender, without this, deciphering the message is nearly impossible.

There was a total of 5 rotors, out of which 3 were chosen for encryption so no of possible rotor combinations were:

$$5 \times 4 \times 3 = 60$$

Each rotor has 26 starting position choices so for 3 rotors.

$$26 \times 26 \times 26 = 17576$$

Now the Plugboard made 10 pairs of letters which means that there are 6 letters which are left over

$$\frac{26!}{6!}$$

Now the order of the 10 pairs is not important, so the possible combinations become.

$$\frac{26!}{10! \times 6!}$$

Now since pair A & B is the same as pair B & A, we can further divide by 2

and since there are 10 pairs, we can divide by 2^{10}

$$\frac{26!}{10! \times 6! \times 2^{10}} = 150{,}738{,}274{,}937{,}250$$

This is how many ways you can connect 20 letters into 10 pairs.

Total is $60 \times 26^3 \times 150{,}738{,}274{,}937{,}250 = 158{,}962{,}555{,}217{,}826{,}360{,}000$ combinations.

B. Flaw in the Enigma Code

Now we will look at the fundamental flaw in the Enigma code that was exploited by Alan Turing and Gordon Welchman.

- German operators were required to transmit certain information every day such as weather reports which were *Wetterbericht* or *Heil Hitler*. These repeated phrases allowed cryptanalysts to search for these specific words and crack the code.
- Despite strict guidelines on how to use the enigma machines, operators sometimes didnt follow the theme due to fatigue. This produced patterns which were used to crack the code.
- The main flaw was that ciphertext and plaintext could never be the same, for example, A could never map to A and B could never map to B.

C. Decryption

Now the allied powers had access to the enigma machine, so they knew how it worked, and the internal circuitry was available to them.

The way they tried to decrypt the code, was to take a text like *Wetterbericht* and pass it through the cipher text to check for what orientation made sure that each letter in *Wetterbericht* was not matched with itself.

For example:

j	x	a	t	g	g	y
	w	c	r	y	b	d

w	e	t	t	e	r	b
	e	r	i	c	h	

in these 2 t's are matched, so then the word is shifted by one.

j	x	a	t	g	g	y
	w	c	r	y	b	d

	w	e	t	t	e	r
	b	e	r	i	c	h

in these 2 r's are matched, so then word is shifted by one and so on.

But this would take a long time.

So, Alan Turing decided to make a machine called the Bombe machine which was named after the Polish

decryption machine Bomba which was used to decrypt Army and Airforce Enigma codes, but it couldn't break the Naval Enigma Code.

The reason the Naval Enigma Code was tougher to break was because the initial rotor setting was sent in an entirely different code. So before starting to break the code, the code for finding rotor positions had to also be broken.

The biggest flaw was that no letter could become itself, and that flaw was fixed by the British. They used the same enigma code but allowed a letter to become itself and called it the *Typex Machine*.

VIII. CASE STUDY: CRYPTOGRAPHY AND ROBOTICS

Cryptography can be used in Robotics, especially in scenarios where privacy, communication, authentication and data integration play a major role.

Cryptography in Robotics is explained below:

A. Workflow Integrity

The integrity of the workflows to be executed is essential in this concept. It must ensure that the workflow as designed by the application developer and signed off by a responsible entity is executed without reductions, additions, or modifications. This includes the flow itself meaning the sequences and branches of actions as well as the parameters of each action. Since e.g., the speed of the robot can dramatically impact the outcome of the robot's action, the integrity of parameters is a crucial part of the workflow's integrity.

The sequence of workflow:

Workflow sequences intended to be executed on the robot system have to be signed by a trusted third party that supports automation of this process i.e., a dedicated signing service. Additionally, the system must be able to verify the signature before the execution is initiated. If the verification fails, the process is aborted in a graceful way without compromising the overall state of the system. This measure not only ensures the integrity of the workflow sequences per se, moreover it allows preventing a client from executing workflow queues, that do not fulfill a set of predefined criteria.

1) *Role of Cryptography in ensuring workflow integrity:* Fulfilling the requirement of authenticity demands to guarantee that the workflow sequence was not signed by the client itself and was signed by a trusted third party. The best way to achieve this is by the implementation of a public key cryptography infrastructure where each entity is initialized with a public key certificate and a corresponding private key. Using this allows us to check whether the information fed into the system originates from the right source or not. In workflow integrity, this means that the trusted party signs the workflow sequence with its private key and the robot system can verify the signature by using the corresponding public-key.

B. Secure Storage

Robots may store sensitive data such as maps, sensor readings, and operational parameters. Cryptography can be used to encrypt this data, protecting it from unauthorized access in case the robot is lost, stolen, or compromised.

1) Security of Robotic Systems: One of the most common and effective ways to secure your robotic system's data is to encrypt it. Encryption is the process of transforming your data into an unreadable format that can only be accessed with a key or a password. Encryption can protect your data from unauthorized access, modification, or leakage, both in transit and at rest. Depending on your needs and preferences, you can use various encryption methods and tools, such as symmetric or asymmetric encryption, public key infrastructure (PKI), or secure sockets layer (SSL).

A few gaps observed are-

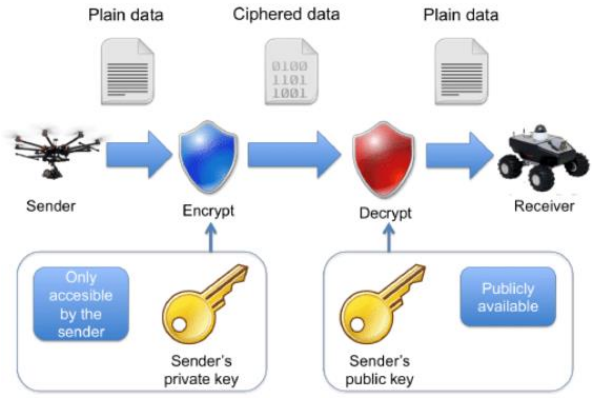
- Lack of major use of public key cryptography in the security of robotic systems.
- Areas like Swarm robotics can better the security with the use of elliptic curve cryptography

A few alternatives are-

1) *Security of robots majorly by the use of public key cryptography:* One of the main obstacles to the robots is security. Majorly robots use techniques like supply chain security for the protection of data and the implementation of cryptographic techniques is minimal. Various cryptographic techniques like public key cryptography can be used to add up to the security. This will allow robots to share their public keys with other systems who want to communicate with them. Therefore, any robotic system in the network can send information to specific robotic addresses, knowing that only the robot that possesses the matching private key can access the data. This would add up a better and extra

layer for the security of the robotic systems making security of the robots an efficient task.

2) *Use of Elliptic curve Cryptography in Swarm Robotics for betterment of their use in the Military:* Swarm Robotics is an approach to coordinating multiple robots as a system consisting of large numbers of mostly simple physical robots. These robotic systems are used in the Military robotic drones which for a robotic network. Various public key cryptography methods are used for the decryption of data in the sector.

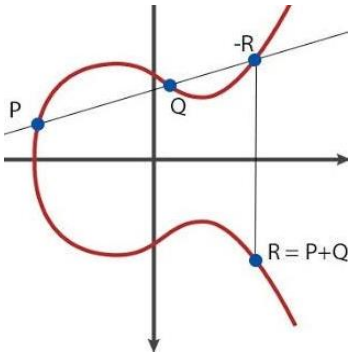


How is ECC a better alternative?

- ECC has a smaller key size which requires very less storage space, which saves the data and storage given the idea that drones have limited storage, this would add up to the life of drones.
- ECC provides a better level of security given the number of solutions possible for a single key using an elliptic curve.

The working behind such a possibility- Let us consider an example of a military drone network consisting of three drones D1, D2 and D3. D1 and D2 belongs to the military network of country A, whose data is encrypted in D1 and D2. D3 belongs to country B which is a spy drone and is trying to decrypt the data from D1 and D2.

If D1 and D2 are protected by symmetric key algorithms, then D3 can decrypt the data of D1 and D2 given the fact that symmetric algorithms can be decrypted easily using a set of combinations.



But if D1 and D2 are protected by elliptic curve cryptography, then a private key and a public key is available. Only D1 and D2 would have access to the private key, D3 would need to decrypt the key in between billions of combinations possible. The reflection of point -R along the X axis will be R. R will be equal to the sum of P and Q, i.e., R=P+Q. This relationship between the points P, Q and R is given by 'dot' which states that

P 'dot' Q =R. Starting from point P, it is dotted with itself for 'n' number of times until the max is reached. Point P is dotted for a secret number of times, i.e., 'n'. In ECC 'n' is considered as the private key which is impossible to find given the number of solutions and combinations possible.

Thus, the data of D1 and D2 cannot be decrypted, when secured with ECC given the possible number of combinations.

XI. Literature Survey

This literature review explores various aspects of cryptography, encompassing cipher techniques, network security, advanced encryption systems, elliptic curve cryptography (ECC), Rivest-Shamir-Adleman (RSA) technique, Diffie-Hellman algorithm, and a case study on the Enigma machine. The review draws insights from a range of scholarly sources and research papers.

1) Cipher Techniques:
   - "Classical Cryptography and Cryptanalysis" by David Kahn provides an extensive historical overview of classical cipher techniques.
   - "Modern Cryptanalysis: Techniques for Advanced Code Breaking" by Christopher Swenson offers insights into modern cryptographic techniques and their analysis.

2) Network Security:
   - "Cryptography and Network Security: Principles and Practice" by William Stallings is a comprehensive reference on cryptographic protocols and network security mechanisms.

- "SSL and TLS: Designing and Building Secure Systems" by Eric Rescorla delves into the design and implementation of SSL/TLS protocols for securing network communications.

3) Advanced Encryption Systems:
   - "Applied Cryptography: Protocols, Algorithms, and Source Code in C" by Bruce Schneier is a seminal work on cryptographic algorithms, including the Advanced Encryption Standard (AES).
   - "Cryptography Engineering: Design Principles and Practical Applications" by Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno provides practical insights into designing and implementing secure cryptographic systems.

4) Elliptic Curve Cryptography (ECC):
   - "Guide to Elliptic Curve Cryptography" by Darrel Hankerson, Alfred Menezes, and Scott Vanstone is a comprehensive guide to understanding and implementing ECC.
   - "Elliptic Curves in Cryptography" by Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart offers an in-depth exploration of the mathematical foundations and applications of ECC.

5) Rivest-Shamir-Adleman (RSA) Technique:
   - "Cryptography: Theory and Practice" by Douglas Stinson covers the theoretical underpinnings of RSA and its practical applications.
   - "Introduction to Modern Cryptography" by Jonathan Katz and Yehuda Lindell provides a rigorous introduction to RSA and other public-key cryptosystems.

6) Diffie-Hellman Algorithm:
   - "Cryptography: An Introduction" by Nigel Smart introduces the Diffie-Hellman key exchange algorithm and its role in modern cryptographic protocols.
   - "Understanding Cryptography: A Textbook for Students and Practitioners" by Christof Paar and Jan Pelzl offers detailed explanations of cryptographic algorithms, including Diffie-Hellman.

7) Case Study: Enigma Machine:
   - "The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography" by Simon Singh provides a captivating account of the Enigma machine and its role in World War II cryptography.
   - "Breaking the Code: Behind the Walls of Chrisitian Science" by Hugh Whitemore offers insights into the personal and technical challenges faced by codebreakers at Bletchley Park during the Enigma decryption efforts.

Incorporating insights from these authoritative sources, this literature review provides a comprehensive overview of cryptography, spanning historical developments to modern cryptographic techniques and case studies.

X. Conclusion

Cryptography and Network security, makes up a fundamental part of our lives. In this increasingly connected world they shape the way we protect of personal data from hackers and other people who seek to access and use our sensitive information.

Over the past years, cryptography has evolved from hill and Caesar ciphers to modern techniques like Advanced Encryption system, Rivest Shamir Adleman method and Elliptic Curve Encryption.

Cryptography has also played a major role in history as explained by the Enigma Case study. Millions of lives were saved because a code was broken. Even today military relies heavily on encryption, when it comes to drone control or even general communication.

With the advent of digital technology, more robust encryption methods which use private and public keys have been introduced. These algorithms use mathematical properties like number theory which provide security which is unbroken by current technology.

As we become more and more interconnected via social media like Instagram, twitter, Whatsapp data security becomes more and more important.

With the arrival of quantum computers, The US congress just passed a new bill, legislating that all agencies must change to new types of cryptography that are “quantum-resistant”. Our current encryption system is very successful because normal computers will take an extremely long time for decryption. But with the advent of quantum computing which proves to be exponentially faster than our modern

supercomputers, new ways of encryption should be made to handle the sheer brute force that quantum computers have.

Hence we conclude that cryptography is an ever-growing field and as the world becomes more interconnected and interdependent, as technology grows. The evolvement of cryptography will also be crucial.

References

[1] GeeksforGeeks. (2024, April 26). Cryptography and its Types. GeeksforGeeks. https://www.geeksforgeeks.org/cryptography-and-its-types/

[2] Neso Academy. (2021, April 17). Cryptography [Video]. YouTube. https://www.youtube.com/watch?v=6_Cxj5WKpIw

[3] GeeksforGeeks, "Hill Cipher," GeeksforGeeks, Jul. 21, 2021. https://www.geeksforgeeks.org/hill-cipher/

[4] D. Sileshi and D. Sileshi, "Playfair Cipher | Baeldung on Computer Science," Baeldung on Computer Science, Mar. 18, 2024. https://www.baeldung.com/cs/playfair-cipher

[5] Wikipedia contributors, "Polyalphabetic cipher," Wikipedia, Mar. 14, 2024. https://en.wikipedia.org/wiki/Polyalphabetic_cipher#:~:text=A%20polyalphabetic%20cipher%20is%20a,fundamentally%20a%20polyalphabetic%20substitution%20cipher

[6] Wikipedia contributors, "Playfair cipher," Wikipedia, Apr. 22, 2024. https://en.wikipedia.org/wiki/Playfair_cipher

[7] Wikipedia contributors, "Letter frequency - Wikipedia," May 04, 2024. https://en.wikipedia.org/wiki/Letter_frequency

[8] Neso Academy, "Cryptography," YouTube. Apr. 17, 2021. [Online]. Available: https://www.youtube.com/watch?v=6_Cxj5WKpIw

[9] Neso Academy, "Cryptanalysis," YouTube. Apr. 27, 2021. [Online]. Available: https://www.youtube.com/watch?v=Kejs-saINOo

[10] Neso Academy, "Brute force attack," YouTube. May 02, 2021. [Online]. Available: https://www.youtube.com/watch?v=DoBqnt7Bf24

[11] Sofia Flynn. (2018, March 29). RSA encryption explained - Proof of RSA [Video]. YouTube. https://www.youtube.com/watch?v=GgePkF0XnRg

[12] David Kahn , "Classical Cryptography and Cryptanalysis"

[13] Christopher Swenson, "Modern Cryptanalysis: Techniques for Advanced Code Breaking"

[14] William Stallings, "Cryptography and Network Security: Principles and Practice"

[15] Eric Rescorla, "SSL and TLS: Designing and Building Secure Systems for securing network communications"

[16] Bruce Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C"

[17] Niels Ferguson, "Cryptography Engineering: Design Principles and Practical Applications"

[18] Darrel Hankerson, Alfred Menezes, and Scott Vanstone, "Guide to Elliptic Curve Cryptography"

[19] Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart, "Elliptic Curves in Cryptography"

[20] Douglas Stinson, "Cryptography: Theory and Practice"

[21] Jonathan Katz and Yehuda Lindell¸ "Introduction to Modern Cryptography"

[22] Nigel Smart, "Cryptography: An Introduction"

[23] Christof Paar and Jan Pelzl, "Understanding Cryptography: A Textbook for Students and Practitioners"

[24] Simon Singh, "The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography"

[25] Hugh Whitemore, "Breaking the Code: Behind the Walls of Christian Science"