Module Code & Module Title

CC5009NI Cyber Security in Computing

Assessment Weightage & Type

40% Individual Coursework 01

Year and Semester

2024 -25 Autumn Semester

Student Name: PRADIP JOSHI

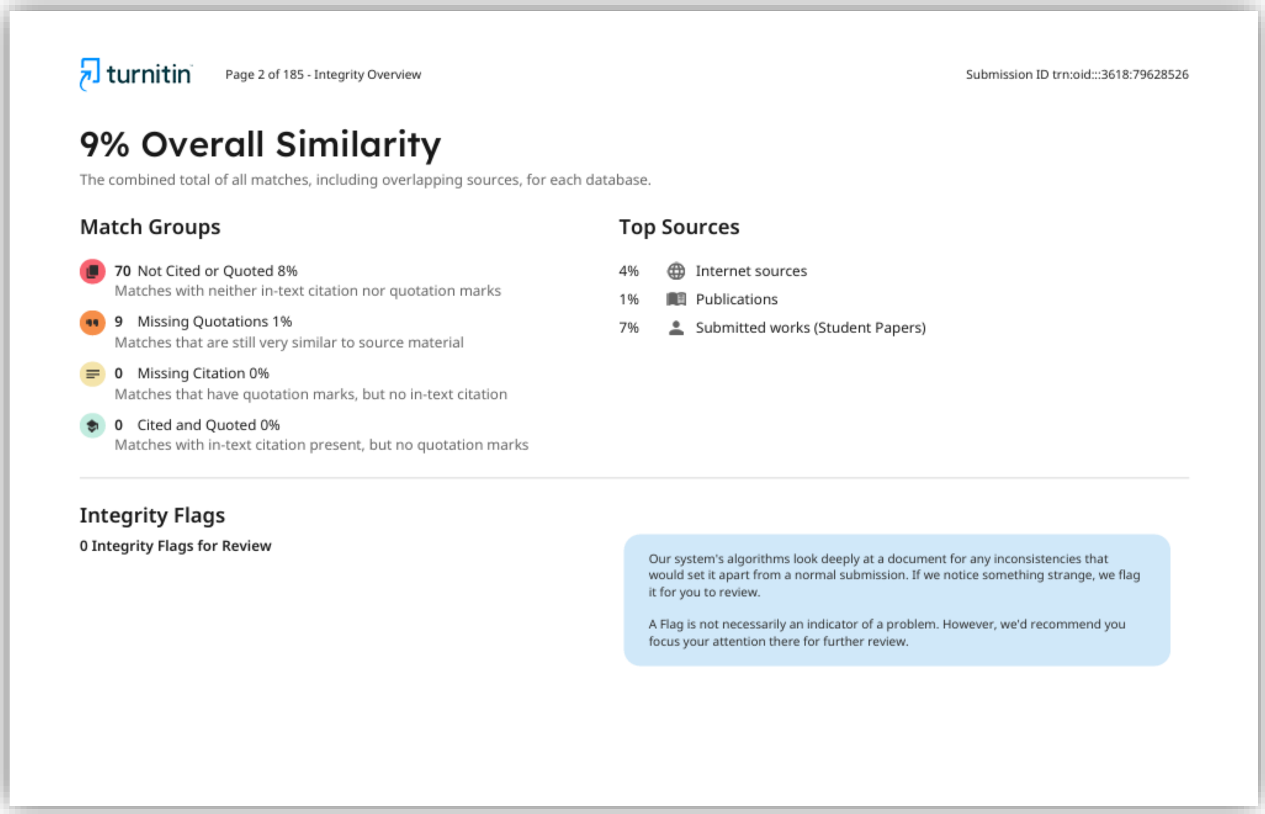
London Met ID: 23047485

College ID: np01nt4a230162

Assignment Due Date: Tuesday, January 20, 2025

Assignment Submission Date: Tuesday, January 20, 2025

Word Count (Where Required): 11884



9% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- **70** Not Cited or Quoted 8%
  Matches with neither in-text citation nor quotation marks
- **9** Missing Quotations 1%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

4%	Internet sources
1%	Publications
7%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Abstract

This project was done to learn about cybersecurity, especially cryptography, and how it helps to protect important information.in today’s world, where technology is used everywhere, keeping information safe is important. The motivation for this project was to understand how encryption works and to create a better method to make information more secure.

The main problem focused on this project was that old encryption methods, like Caesar cipher, are easy to break using modern techniques. So, it was decided to modify the Caesar cipher to make it more secure and harder to break.

This project was started by learning about the basics of IT security, the CIA triad, and several types of encryptions. The Caesar cipher was studied in detail to understand its strengths and weaknesses. Then an advanced version of Caesar cipher was created by adding new features like random key generation, converting letters into binary, and using XOR and permutation operations. The new algorithm was tested with different examples and the results were analysed.

After finishing the project, it was found that the new cipher was much stronger than the original Caesar cipher and could resist simple attacks like frequency analysis. This project helped to understand how to make encryption methods better and how cryptography can solve security problems.

This project is important in the field of information security because it shows how the old methods can be improved to manage modern challenged. This project not only improved the theoretical knowledge but also gave the practical experience in designing and testing cryptographic algorithm. It also teaches the value of learning and experimenting with new ideas to make systems more secure. This project shows the ongoing need for secure systems and the significant role of cryptography plays in protecting data in an increasingly digital world.

Contents

Table of Figures

1. Introduction

we are living in an era where Information technology plays a vital role, and data has become a crucial asset for any organization, business or any other fields striving for success in their respective field of interest. As the value of data increases, so does the risk associate to it. It is important for every organization or business to keep their data safe from various threats. Protecting the data is important to maintain CIA (Confidentiality, Integrity, and Availability) triad. This triad is the foundation of any effective information security strategy. Various techniques and tools are employed to maintain CIA triad and Cryptography is one of the most significant techniques that can effectively help to maintain CIA triad.

The concept of IT security along with its types is properly discussed in this report. The main components or core pillar of Information security, (CIA) triad is also discussed. This report explores about the concept of cryptography, history of cryptography along with its types (Symmetric and Asymmetric) and examples. For practical illustration, Caesar cipher is selected for analysis, and its advantages and disadvantages are discussed. By doing necessary modifications a new cryptography algorithm is created that enhances the security compared to Caesar cipher. Furthermore, the strengths and weaknesses of the newly created algorithm is created is also analysed.

This report was prepared as the result of research which was performed to provide a comprehensive understanding of core concepts of IT security and cryptography. Creating a new cryptographic algorithm enhances the learning by applying the theoretical knowledge to develop a more secure encryption method. To complete this coursework, we need to critically evaluate existing cryptographic systems and innovate new solutions, which does a great help in enhancing the knowledge in the field of cyber security.

1.1. Aims and objectives.

The aim of this coursework is to design a new cryptography algorithm by making necessary modifications on already existing algorithm to enhance its security. The objectives are:

- To provide detailed explanation of IT security.
- To explore about cryptography, its types, examples, and history in detail.
- To select a cryptographic algorithm and research about it.
- To create a new, more secured cryptographic algorithm by making changes in the selected algorithm.
- To analyse applications and effectiveness of the newly created algorithm.

1.2. Information Technology (IT) security

IT security is a term that describes the set of strategies, techniques, tools, and solutions that are used to protect the Confidentiality, Integrity, and Availability (CIA) triad of any organization's assets and data.

A thorough IT security plan uses both human resources and technologies to prevent, identify and address many IT risks. Almost all aspects of companies have moved online in the last ten years. This has increased the likelihood that any firm might become the target of a cyberattack, which could aim to steal confidential data, including payment and customer information, trade secrets, or intellectual property, or just to damage the company's reputation. Furthermore, the increasing number of linked devices, the shift to the cloud, and the growing popularity of remote have given hackers and other cyber-criminals almost infinite options for starting an attack.

To protect cloud-based assets, companies must update and reinforce their security procedures due to the increased attack surface and the increasing complexity of digital adversaries. (agoffe, 2021)

1.2.1 Types of IT security

1. **Network security**

   Network security is the domain of techniques and technologies that keeps internal networks safe from intrusions and data breaches (cloudflare, 2024). It is a collection of guidelines and configurations intended to maintain the confidentiality, integrity and availability of network and data utilizing both hardware and software technologies. Regardless of size, industry, or architecture, all organizations need to have some level of network security solutions to protect themselves from constantly evolving array of cyberthreats that exist in the world today. (forcepoint, 2024) Network security can be maintained by using the tools and techniques such as: Access control, User authentication, Firewalls, DDoS (Distributed Denial of Service) protection, Browser isolation.

2. **Information security**

   Protecting sensitive data from unwanted access, disclosure, use, alteration, or interruption is known as information security. It assists in guaranteeing that sensitive organizational data is accessible to authorizes users while maintaining its confidentiality and integrity. (Jim Holdsworth, 2024) Technologies like cloud access security brokers (CASB), deception tools, endpoint detecting and response (EDR) and security testing for DevOps are used to maintain information security (Microsoft, 2024).

3. **Operational security**

   Operational security is a technique that finds harmless activities that might unintentionally give a cybercriminal access to sensitive or valuable information. It encourages IT and security managers to consider their systems and activities from the viewpoint of an attacker. It encompasses analytical procedures and activities such as social media monitoring, behaviour tracking, and security best practices. (fortinet, 2024)

4. **Physical security**

   Physical security means to protect against theft, damage, and unauthorized access to people, property, and data. In contrast to digital cousin, it functions in the actual world and uses a variety of tactics to establish safe spaces.

   Implementing access control, intrusion detection systems, perimeter security, and surveillance are some of the tactics that can be used to maintain physical security. (Burge, 2023)

1.3. CIA (Confidentiality Integrity and Availability) triad

The CIA triad stands for Confidentiality, Integrity, and Availability. It is an extremely popular model that servers as the foundation for the security systems in the CIA triad. It is used to identify weaknesses and develop solutions. Information confidentiality, integrity and availability are essential to a business's operations, and the CIA triangle divides these three concepts into distinct focal points. This distinction is useful because it directs security teams as they identify the many approaches, which can be used to solve each issue. (Fortinet , 2024)

Each component of CIA triad is described below:

1. **Confidentiality**

   The ethical and legal requirement to protect confidential data from unauthorized access or disclosure is known as confidentiality. Maintaining confidentiality involves taking precautions to prevent unauthorized people from accessing data. This implies that only officially authorized people or organizations should have access to sensitive information. A company needs to protect the privacy of its financial data by making sure that only authorized users can access it. Protecting confidential information from unauthorized access, disclosure or exposure is another definition of confidentiality. This approach is intended to ensure the privacy of sensitive data by limiting access to specific data to those who are legally authorized. Data breaches and unauthorized access can cause reputation and legal repercussions to the organization. The implementation of access controls and encryption algorithms like Advanced Encryption Standard (AES), are essential for maintaining the principal confidentiality within network. (Osazuwa, 2023)

   Examples of confidentiality are found in a variety of access control techniques, such as password less sign-on, two-factor authentication, and others. However, confidentiality is more than just granting access to authorized users, it also involves preventing access to specific files. By using encryption, organizations can protect their data from intentional as well as unintentional disclosure. (Fasulo, 2024)

2. **Integrity**

   Integrity is the safeguarding of data against unauthorized modifications or changes. This requires that any modifications must be noted and recorded, and information should never be changed without the appropriate authorization. Maintaining the integrity of business's product designs is essential, making sure that any changes are made only with the proper authorization. Maintaining data's accuracy, consistency, and dependability across the course of its lifecycle is another aspect of integrity. Integrity can be ensured by preventing malicious actors from modifying, erasing, or interfering with data to ensure data integrity. Since any alteration to the original content can cause detectable change in the hash value, hash functions and digital signatures are widely used cryptographic techniques for confirming data integrity. Preventing data integrity is essential to stop the spread of inaccurate or tainted information, which might have serious consequences in crucial industries like healthcare and finance. (Osazuwa, 2023)

In addition to encryption and access control, there are numerous other methods to safeguard data integrity against corruption and attacks. There are instances when a read-only file is sufficient. Data checksums and hashing are sometimes utilized, enabling data audits to make sure the data has not been hacked. (Fasulo, 2024)

3. Availability

The idea of availability refers to the guarantee that information may be readily accessed by authorized personnel when required. This implies that it is important to protect information systems from disruptions that can restrict user's access to data. Availability refers to the assurance of the network resources, services, and data's usability, particularly when those with the appropriate authorization needs them. Either technical issues or cyberattacks have the potential to disrupt organizational procedures that can cause monetary loss and affect the organization's reputation in the public eye. High availability architectures and failover techniques are essential for ensuring continuous access to network services. (Osazuwa, 2023)

For example, distributed denial of service (DDoS) assaults depends on limited availability. To ensure availability, redundancy should be implemented in the systems and a DDoS response plan should be developed. However, even in the absence of an attack, systems may malfunction and become unavailable. So, fault tolerance and load balancing can be effective ways to prevent system from failure. (Fasulo, 2024)

1.4. Cryptography

Cryptography is the process of converting plain text into cyphertext. The plaintext is converted into ciphertext by the sender. The cipher text is then sent to receiver. After receiving the cyphertext, the authorized recipient decrypts it and returns it to its original format. The primary goal of cryptography is to prevent unauthorized access to the data.

Encryption is the process of masking plain text so that its original form is concealed. The unreadable text that is generated after encryption is called cipher text. The process of changing the cipher text into original plain text is called decryption. Cryptosystems are the entities that offers encryption and decryption. Cryptography is essential to the security component. Nowadays cryptography is widely used because it offers numerous security objectives to guarantee data confidentiality. (S.Suguna, 2016)

1.5. History of Cryptography

The content below gives the chronological history of cryptography.

1. **1900 BC: Egyptian Cryptography**
    - The earliest example of cryptography comes from an Egyptian scribe who used non-standard hieroglyphs in an inscription. These altered symbols disguised the true meaning of the message, marking the first recorded instance of written encryption.
2. **1500 BC: Assyrian Intaglio Seals**
    - Assyrian merchants employed intaglio seals, flat stones engraved with images and writing, as authentication tools. These functioned as early digital signatures, allowing only the holder to produce the unique imprint for trade transactions.
3. **500-600 BC: ATBASH Cipher in Hebrew**
    - Hebrew scribes used the ATBASH cipher, a reversed-alphabet substitution cipher, to encode parts of the Book of Jeremiah. This simple yet effective method protected religious and historical texts.

4. **487 BC: Greek Skytale**

    - The Greeks developed the skytale, an early transposition cipher. Messages were written on a leather strip wrapped around a staff. The recipient, with a staff of the same diameter, could decode the message by rewrapping the strip.

5. **100-44 BC: Caesar Cipher**

    - Julius Caesar introduced a substitution cipher where letters in the alphabet were shifted by a fixed number. For example, an "A" became "D." This provided a simple but effective way to secure government communications.

6. **725-790 AD: Al-Khalil’s Contributions**

    - The Arab scholar Al-Khalil authored a cryptographic book inspired by solving a Greek cryptogram. He introduced techniques like known plaintext analysis, which became foundational for cryptanalysis.

7. **1379: Nomenclator Systems**

    - Gabrieli di Lavinde created a nomenclator system combining substitution ciphers and small codes for Clement VII. This method became standard in diplomacy for 450 years despite the invention of stronger systems.

8. **1466: Alberti’s Polyalphabetic Cipher**

    - Leon Battista Alberti invented the first polyalphabetic cipher and a cipher disk to simplify encoding. His work marked a significant leap in cryptographic strength and versatility, remaining unbroken for centuries.

9. **1518: Trithemius’ Steganographic Ciphers**

    - Johannes Trithemius published the first cryptographic book. He introduced polyalphabetic ciphers and steganographic methods where prayers concealed encoded messages.

10. **1553: Belaso's Passphrase Cipher**

- Giovan Batista Belaso advanced cryptography by using passphrases as keys for polyalphabetic ciphers, a method often misattributed to Blaise de Vigenère.

11. **1585: Vigenère Cipher**

- Blaise de Vigenère expanded on polyalphabetic techniques by introducing ciphertext and plaintext autokey systems, making messages significantly harder to decipher.

12. **1790: Jefferson Wheel Cipher**

- Thomas Jefferson invented the wheel cipher, an encryption device using rotating disks. Its principles influenced later cryptographic devices like the WWII strip cipher.

13. **1623: Bacon's Biliteral Cipher**

- Sir Francis Bacon developed a biliteral cipher using variations in typeface to encode messages. This innovation foreshadowed binary encoding.

14. **1917: William F. Friedman**

- Considered the father of U.S. cryptanalysis, Friedman conducted groundbreaking work in cryptographic analysis and helped establish formal methods for breaking codes.

15. **1933-1945: The Enigma Machine**

- Nazi Germany's Enigma machine revolutionized encryption during WWII. Although highly complex, it was deciphered by Polish mathematicians and further exploited by British cryptanalysts like Alan Turing at Bletchley Park. Their work laid the foundation for modern computing and cryptanalysis.

16. **1976: Data Encryption Standard (DES)**

- IBM's DES, based on the Lucifer cipher, became a global encryption standard. Despite its eventual vulnerabilities, it established the framework for modern block ciphers.

17. 1976: Diffie-Hellman Key Exchange

- Whitfield Diffie and Martin Hellman introduced public-key cryptography, allowing secure communication without pre-shared keys. This breakthrough addressed the challenge of key distribution.

18. 1977: RSA Algorithm

- Ronald Rivest, Adi Shamir, and Leonard Adleman developed RSA, the first practical public-key encryption system. Based on the difficulty of factoring large numbers, RSA remains a cornerstone of secure communication.

19. 1991: Pretty Good Privacy (PGP)

- Phil Zimmermann released PGP to provide the public with access to high-security cryptography. Its availability as freeware made it a widely adopted standard for email encryption.

20. 1994: RC5 Algorithm

- Ron Rivest introduced RC5, a flexible encryption algorithm allowing users to customize parameters like block size and key length, offering tailored security solutions. (Jackob, 2021)

1.6. Symmetric and Asymmetric encryption

1.6.1. Symmetric encryption

Symmetric encryption refers to encryption techniques that uses the same secret key to encrypt and decrypt. Symmetric encryption techniques rely on mathematical operations that can be programmed into incredibly fast computational algorithms, allowing even small computers to efficiently complete the encryption and decryption processes. One of the major drawbacks of symmetric algorithm is that both sender and the recipient must have the secret key. If the key ends up in unauthorized or malicious user, they can decrypt the message, potentially without the sender or receiver realizing that the communication is compromised. (Michael E. Whitman, 2012) There are two types of symmetric algorithms. They are:

1. **Block algorithms**
   Blocks of electronic data are encrypted using block algorithm. The designated bit-lengths of the data are modified while continuing to use the same designated key. Then every block uses this key. While waiting for the full blocks, the encryption system stores the data in its memory components when encrypting network stream data. The system's waiting time may compromise data security and integrity result in a security flaw. To solve this problem the block of data can be decreased and merged with preceding encrypted data block contents, till all the remaining blocks comes.
2. **Stream algorithms**
   Stream algorithms arrive in data stream algorithms rather than being stored in the memory of the encryption system. Since a disk or system does not stores data without encryption in the memory components, this process is viewed as more secured. (Teach Computer Science, 2024)

Examples of symmetric encryption algorithms:

- AES (Advanced Encryption Standard)
- DES (Data Encryption Standard)
- IDEQA (International Data Encryption Algorithm)
- RCP (Rivest Cipher 4)
- RC5 (Rivest Cipher 5) (cryptomathis, 2024)

1.6.2. Asymmetric encryption algorithm

Two separates but connected keys where one key is used to encrypt the message, and another is used to decrypt the message in asymmetric algorithm. Consider first key as 'A' and second key as 'B,' here if the key 'A' is used for encryption then only the key 'B' can decrypt the message and vice versa. Asymmetric encryption offers sophisticated answers to the problems related to confidentiality and verification issues. This method works best when one key is used as a private key, which is kept secret and is disclosed only to the owner like symmetric key algorithm and the other key is used as a public which is kept in a place where anybody can access it. Because of this reason asymmetric encryption is also commonly known as public-key encryption. (Michael E. Whitman, 2012) Some examples of asymmetric encryption are:

- RSA (Rivest Shamir Adleman)

- Digital Signature Standard
- ELGAMAL SIGNATURE
- DIFFIE HELLMAN KEY EXCHANGE (S.Suguna, 2016)

2. Background

2.1. Caesar cipher

Julius Caesar created the Caesar cipher which is a simple encryption method to communicate with his supports in secret. It operates by "shifting" or "Keying" the plaintext message's letters by a certain number of positions. (Satish Dadasaheb Tribhuvan, 2024) It is one of the oldest methods of encryption that was used by Julius Caesar and other military leaders during the military operations to protect their communication. The feature of Caesar cipher like simplicity and easiness in using it helped to create the foundation for today's world's advanced cryptographic methods. It is a symmetric algorithm that means single key is used to perform tasks like encryption and decryption. (splunk, 2024)

2.2. Example of Caesar cipher

Caesar cipher has given an expression that is written as:

C = (P + 3) mod 26

Where,

C = ciphertext

P = Plaintext

Consider the word 'APPLE' for encryption. We need to apply the expression of Caesar cipher.

Here,

For A,

C = (A + 3) = D

For P,

C = (P + 3) = S

For P,

C = (P + 3) = S

For L,

C = (L + 3) = O

For E,

C = (E + 3) = H

Therefore, encrypted form of ‘APPLE’ using Caesar cipher is ‘DSSOH.’

Now,

Decrypting the cipher text

For D,

P = (D -3) = A

For S,

P = (S – 3) = P

For S,

P = (S – 3) = P

For O,

P = (O – 3) = I

For H,

P = (H -3) = E

Therefore, decrypted form of ‘DSSOH’ is ‘APPLE.’

2.3. Advantages and Disadvantages of Caesar cipher algorithm.

1. **Advantages**
    - Easy to put into practice.
    - Most basic cryptographic method.
    - A single short key is used for encryption and decryption.
    - It is the ideal approach for a system that does not uses sophisticated coding techniques.
    - Needs a small amount of processing power. (javatpoint, 2024)
    - Does a significant help to beginners to learn encryption (splunk, 2024).

2. **Disadvantages**
    - Easily crackable.
    - Provides little security.
    - The complete message can be deciphered by examining the letter pattern. (javatpoint, 2024)
    - It is not suitable for secure communication because its simplicity makes it easy to break.
    - No message Confidentiality, Integrity, and authenticity. (splunk, 2024)

3. Development

This section focuses on the development of a new cryptographic algorithm specifically designed to overcome the limitations of the Caesar cipher. While the Caesar cipher has been a foundational encryption technique, it suffers from several vulnerabilities, primarily its simplicity and the limited key space, making it susceptible to frequency analysis and brute-force attacks. The new algorithm addresses these weaknesses by introducing more complex encryption methods.

3.1. Advanced Caesar Cipher

The Advanced Caesar Cipher algorithm is a symmetric encryption and decryption process that utilizes bitwise operations to transform plaintext into ciphertext and vice versa. During encryption, the plaintext is right shifted by a random set of numbers created with respect to the positions (e.g., the first letter is shifted by the first number of the set). The binary representation of the plaintext then undergoes an Expansion and Permutation (E/P) to increase complexity. Afterward, the bits are divided into two equal halves of five bits, with each half left-shifted by three bits. The shifted halves are combined and XORed with a constant, followed by a permutation (P10) to further mix the bits. Finally, the permuted bits are divided into equal halves of five bits and mapped back to alphabetic values, creating the ciphertext.

During decryption, the ciphertext is divided into 10-bit blocks, and an inverse permutation ($P10^{-1}$) is applied before XORing with the constant. The bits are then divided into equal halves of five bits and right shifted by three bits, followed by an inverse E/P to restore the original bit structure. The binary values are converted back into alphabetic characters and are left-shifted by the corresponding number from the random set. Specifically, if a letter is in the first position, it is left-shifted by the first number in the random set; if it is in the second position, it is left-shifted by the second number, and so on, to recover the original plaintext.

This combination of shifting, XOR operations, and permutation ensures strong security for the data.

3.1.1. Encryption and Decryption flowchart

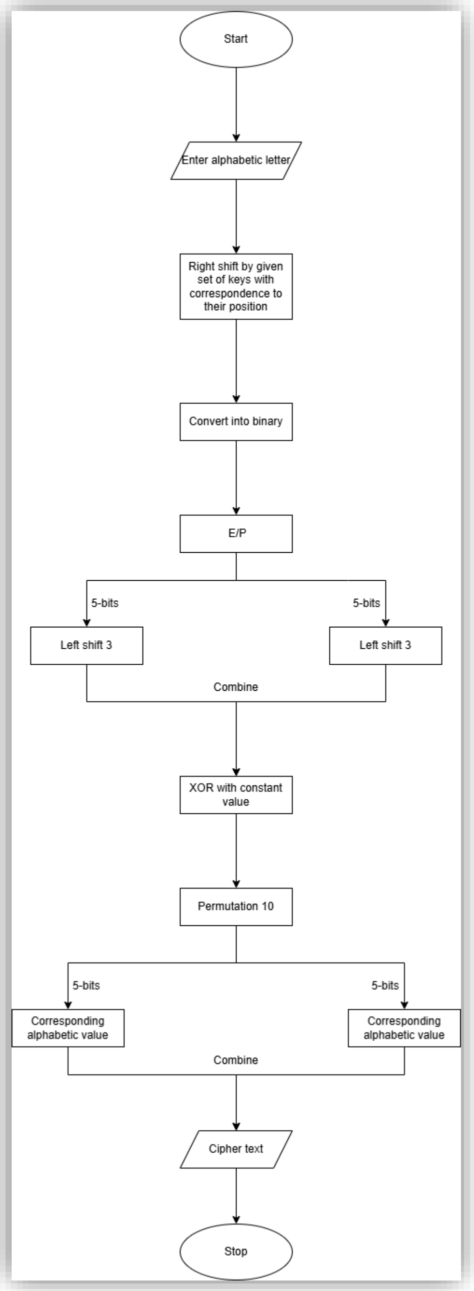


Figure 1 : Flowchart for encryption process.

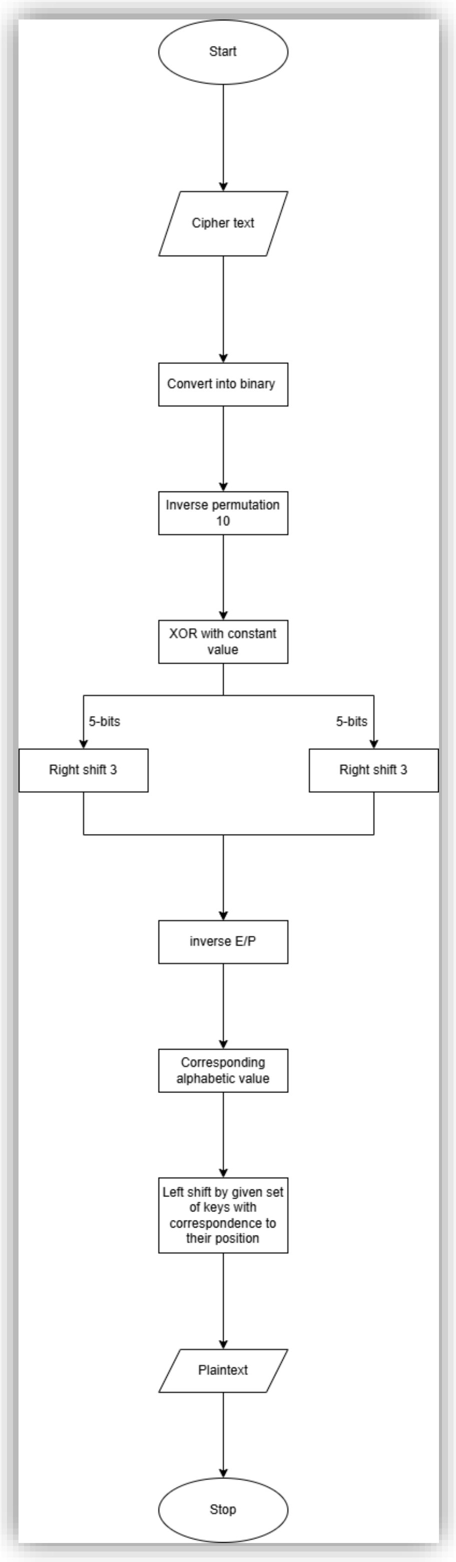


Figure 2 : Flowchart of Decryption process.

3.1.2. New operations added.

1. Key generation

   In the standard Caesar cipher, the shift value is static, often with the shift value of 3. The key generation process is modified by selecting a set of random numbers to serve as the shifting key for each letter in the plaintext to enhance security and reduce predictability. The selected set of random numbers is:

   (10, 8, 12, 6, 18, 7, 13, 5).

   Once the numbers are exhausted, they are reused in a repeating cycle. This modification introduces more variability to the encryption process, making it more resistant to traditional cryptanalysis methods such as frequency analysis.

2. Conversion into binary

   After the plaintext are shifted using the generated key, the resulting letters are converted into binary format. Specifically, each letter is represented as a 5-bit binary value. This conversion is necessary to perform cryptographic operations, such as permutation and shifting, which are easier to perform at a bit level.

3. Expand and Permutate (E/P)

   Once the letters are converted into binary, the next operation is expansion and permutation (E/P). in this step, the binary values are expanded from 5-bits to 10-bits applying custom E/P table. This expansion helps increase the complexity of the cipher and enhances security by making the ciphertext less predictable. The E/P table I created is as follows:

   E/P = [2, 3, 5, 4, 3, 2, 1, 5, 1, 4].

   This table rearranges the bits to increase the cipher's complexity.

4. Left Shift by three

   The expanded binary output from the previous step is divided into two halves, each containing 5-bits. A left shift by three is then performed on both halves. This step further complicates the encryption process and adds another layer of complexity.

5. XOR

   The output Left shift is then XORed with the constant that was created which is 1 1 0 0 1 1 1 0 0 0.

6. Permutation 10 (P10)

   After performing the XOR operation, the output is subjected to a final permutation (P10). This step applies a predefined table to rearrange the bits in a specific order, enhancing the complexity of the ciphertext. The P10 table used is:

   P10 = [10, 5, 6, 3, 2, 4, 1, 8, 9, 7]

   This step ensures that the final ciphertext is sufficiently scrambles, making it harder to break without the correct decryption key.

7. Inverse Permutation 10 (Inverse P10)

   While decrypting P10 needs to be reversed. This is done to return scrambled bits to their correct order, allowing for the correct decryption of the ciphertext. The Inverse P10 table is given below:

   $P10^{-1}$ = [7, 5, 4, 6, 2, 3, 10, 8, 9, 1]

8. Right shift by three

   After inverse P10 the binary output is divided into two halves, each containing 5-bits. A right shift by three is then performed on both halves. This step takes more closer to the original bit.

9. Inverse E/P

   While decrypting the E/P needs to be reversed. To do so an inverse E/P table is created that plays essential role in decryption. The inverse E/P table of the E/P table is:

   E/P^{-1} = [7, 6, 2, 4, 3]

3.1.3. Data needed for the new algorithm.

1. The selected set of random numbers for shifting in first step.

   Set = [10, 8, 12, 6, 18, 7, 13, 5]

2. Binary conversion table for alphabets.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z		!	.	?	*	£
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Convert these numbers to 5-bit binary sequences.

A = 0 = 00000	B = 1 = 00001	C = 2 = 00010	D = 3 = 00011
E = 4 = 00100	F = 5 = 00101	G = 6 = 00110	H = 7 = 00111
I = 8 = 01000	J = 9 = 01001	K = 10 = 01010	L = 11 =01011
M = 12 = 01100	N = 13 = 01101	O = 14 = 01110	P = 15 =01111
Q = 16 = 10000	R = 17 = 10001	S = 18 = 10010	T = 19 =10011
U = 20 = 10100	V = 21 = 10101	W = 22 = 10110	X = 23 =10111
Y = 24 = 11000	Z = 25 = 11001	space = 26= 11010	! = 27 = 11011
. = 28 = 11100	? = 29 = 11101	*= 30 = 11110	£ = 31 =11111

3. Expand and Permutate (E/P) table

E/P = [2, 3, 5, 4, 3, 2, 1, 5, 1, 4]

4. XOR constant

XOR constant = 1 1 0 0 1 1 1 0 0 0

5. Permutation 10 (P10)

P10 = [10, 5, 6, 3, 2, 4, 1, 8, 9, 7]

6. Inverse E/P

E/P^{-1} = [7, 6, 2, 4, 3]

7. Inverse Permutation 10 (Inverse P10)

$P10^{-1}$ = [7, 5, 4, 6, 2, 3, 10, 8, 9, 1]

3.1.4. New Encryption and Decryption algorithm

1. Encryption algorithm

Step 1: Shift the letter by the given set of random numbers.

Step 2: Convert the letters into binary by using values from table.

Step 3: Perform E/P to the binary form of letters using given table.

Step 4: Divide expanded bits into two equal halves and left shift each half by three.

Step 5: XOR the shifted value with given constant.

Step 6: Perform Permutation 10 with XORed value.

Step 7: Divide all the bits into equal halves of five bits and write the alphabetic value according to the corresponding binary value.

Example of Encryption

Encrypting ‘OK.

Step 1: Shift O and K by 10 and 8, respectively.

O + 10 = Y

K + 8 = S

Step 2: Convert Y and S into binary by using values from table.

Y = 11000

S = 10010

Step 3: Perform E/P to binary values of Y and S

E/p for 11000,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	1	0	0	0	0	1	1	0	1	0

E/P for 10010,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	0	0	0	1	0	0	1	0	1	1

Therefore,

E/P of 11000 = 1000011010

E/P of 10010 = 0001001011

Step 4: Divide the expanded bits into two halves and left shift each half by three.

Given bits are: 1000011010 and 0001001011

Actual bits	1000011010		0001001011	
Before shift	Left half	Right half	Left half	Right half
	10000	11010	00010	01011
After shift	00100	10110	10000	11010

Therefore, after shift,

1000011010 = 0010010110

0001001011 = 1000011010

Step 5: XOR both values with 1 1 0 0 1 1 1 0 0 0

For 0010010110

Actual value	0	0	1	0	0	1	0	1	1	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	1	1	0	1	0	1	1	1	0

Output = 1110101110

For 1000011010

Actual value	1	0	0	0	0	1	1	0	1	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	1	0	0	1	0	0	0	1	0

Output = 0100100010

Therefore,

XOR of 0010010110 = 1110101110

XOR of 1000011010 = 0100100010

Step 6: Perform Permutation 10 with the outputs of Step 5

For 1110101110,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	0	1	0	1	1	0	1	1	1	1

Output = 0101101111

For 0100100010,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	0	1	0	0	1	0	0	0	1	0

Output = 0100100010

Therefore,

P10 of 1110101110 = 0101101111

P10 of 0100100010 = 0100100010

Step 6: Divide the bits into equal parts each of five bits and write the alphabetic value according to the corresponding binary value.

From table,

01011 = L

01111 = P

01001 = J

00010 = C

Therefore, encrypted form of ‘OK’ = LPJC

Decryption steps

Step 1: Convert the cipher text into binary by using the value from the table and combine them.

Step 2: Divide the cipher text into equal parts of 10-bits and perform inverse Permutation 10 ($P10^{-1}$).

Step 3: XOR each block with the XOR constant.

Step 4: Divide XORed values into two equal halves and right shift by three.

Step 5: Perform inverse Expand and Permutate (E/P^{-1})

Step 6: Conver the binary value into Alphabetical value.

Step 7: Left shift the letters using the set of random numbers.

Example of Decryption

Here, ciphertext = LPJC

Step 1: Convert cipher text into binary from the table and combine them.

L = 01011

P = 01111

J = 01001

C = 00010

After combining, cipher text = 01011011110100100010

Step 2: Divide the cipher text into equal parts each of 10-bits and perform inverse permutation 10 ($P10^{-1}$).

First half = 0101101111

Second half = 0100100010

Inverse P10 each half ($P10^{-1}$)

For 0101101111,

$P10^{-1}$ table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	1	1	1	0	1	0	1	1	1	0

For 0100100010,

$P10^{-1}$ table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	0	1	0	0	1	0	0	0	1	0

Therefore,

$P10^{-1}$ of 0101101111 = 1110101110

$P10^{-1}$ of 0100100010 = 0100100010.

Step 3: XOR each value with XOR constant (1 1 0 0 1 1 1 0 0 0)

For 1110101110,

Actual value	1	1	1	0	1	0	1	1	1	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	0	1	0	0	1	0	1	1	0

Output = 0010010110

For 0100100010,

Actual value	0	1	0	0	1	0	0	0	1	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	0	0	0	0	1	1	0	1	0

Output = 1000011010

Therefore,

XOR of 1110101110 = 0010010110

XOR of 0100100010 = 1000011010

Step 4: Divide both XORed values into two halves and right shift by three.

Given bits are: 0010010110 and 1000011010

Actual bits	0010010110		1000011010	
Before shift	Left half	Right half	Left half	Right half
	00100	10110	10000	11010
After shift	10000	11010	00010	01011

Therefore, after shift

1000011010 = 1000011010

0001001011 = 0001001011

Step 5: Perform inverse E/P (E/P^{-1}) to the output of step 3.

For 1000011010,

E/P^{-1} table	7	6	2	4	3
Arrangement of bits	1	1	0	0	0

For 0001001011,

E/P^{-1} table	7	6	2	4	3
Arrangement of bits	1	0	0	1	0

Therefore,

E/P^{-1} of 1000011010 = 11000

E/P^{-1} of 0001001011 = 10010

Step 6: Convert binary value into Alphabet by using values from table.

11000 = Y

10010 = S

Step 7: Left shift the letters using the set of random numbers.

Y left shift by 10 = O

S left shift by 8 = K

Therefore, Decrypted form of LPJC = OK

3.1.5. Why the modification was necessary.

Caesar cipher is the simplest cryptographic algorithm that can easily decrypted. So, to make it more secure some modifications were done. The modifications along with their significance are given below:

1. Key Generation

   This modification prevents the simplicity and predictability of a static Caesar cipher by introducing variability that increases security.

2. Conversion into Binary

   Converting to binary allows us to work with the message at a lower level. We can perform bitwise operations that enhances security through complexity.

3. Expand and Permutate (E/P)

This expansion and permutation increase the disorderliness of the data and ensure that each bit in the final output is influenced by multiple bits from the input that makes the encryption more secure.

4. Left shift by three.

The left shift introduces additional complexity by changing the position of the bits in the sequence, further complicating the relationship between the ciphertext and the original message.

5. XOR with constant

The XOR operation helps to make the data more random and introduces non-linearity that makes it harder to crack the algorithm.

6. Permutation 10

It increases the complexity of the algorithm by rearranging the bits again. The rearrangement makes it more difficult for attackers to analyse or perform the reverse-engineering.

7. while decrypting inverse permutation 10 and inverse Expansion and Permutation is done in place of Permutation 10 and Expansion and Permutation, as we need to reverse the process of encryption while decrypting the cipher text.

4. Testing

In this section, the working steps of the algorithm developed in TASK 03 are demonstrated by applying it to five different examples. These examples highlight how the algorithm converts plain text into cipher text and vice versa.

4.1. Test one

4.1.1. Encrypting "NO"

Step 1: Shift N and O by 10 and 8, respectively.

N + 10 = X

O + 8 = W

Step 2: Convert X and W into binary by using values from table.

X = 10111

W = 10110

Step 3: Perform E/P to binary values of X and W

E/p for 10111,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	0	1	1	1	1	0	1	1	1	1

E/P for 10110,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	0	1	0	1	1	0	1	0	1	1

Therefore,

E/P of 10111 = 0111101111

E/P of 10110 = 0101101011

Step 4: Divide the expanded bits into two halves and left shift each half by three.

Given bits are: 0111101111 and 0101101011

Actual bits	0111101111		0101101011	
Before shift	Left half	Right half	Left half	Right half
	01111	01111	01011	01011
After shift	11011	11011	11010	11010

Therefore, after shift,

0111101111 = 1101111011

0101101011 = 1101011010

Step 5: XOR both values with 1 1 0 0 1 1 1 0 0 0

For 1101111011

Actual value	1	1	0	1	1	1	1	0	1	1
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	0	0	1	0	0	0	0	1	1

Output = 0001000011

For 1101011010

Actual value	1	1	0	1	0	1	1	0	1	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	0	0	1	1	0	0	0	1	0

Output = 0001100010

Therefore,

XOR of 1101111011 = 0001000011

XOR of 1101011010 = 0001100010

Step 6: Perform Permutation 10 with the outputs of Step 5

For 0001000011,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	1	0	0	0	0	1	0	0	1	0

Output = 1000010010

For 0001100010,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	0	1	0	0	0	1	0	0	1	0

Output = 0100010010

Therefore,

P10 of 0001000011 = 1000010010

P10 of 0001100010 = 0100010010

Step 6: Divide the bits into equal parts each of five bits and write the alphabetic value according to the corresponding binary value.

From table,

10000 = Q

10010 = S

01000 = I

10010 = S

Therefore, encrypted form of ‘OK’ = QSLS

4.1.2. Decrypting "QSIS"

Here, ciphertext = QSIS

Step 1: Convert cipher text into binary from the table and combine them.

Q = 10000

S = 10010

I = 01000

S = 10010

After combining, cipher text = 10000100100100010010

Step 2: Divide the cipher text into equal parts each of 10-bits and perform inverse permutation 10 ($P10^{-1}$).

First half = 1000010010

Second half = 0100010010

Inverse P10 each half ($P10^{-1}$)

For 1000010010,

$P10^{-1}$ table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	0	0	0	1	0	0	0	0	1	1

For 0100010010,

$P10^{-1}$ table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	0	0	0	1	1	0	0	0	1	0

Therefore,

$P10^{-1}$ of 0101101111 = 0001000011

$P10^{-1}$ of 0100100010 = 0001100010

Step 3: XOR each value with XOR constant (1 1 0 0 1 1 1 0 0 0)

For 0001000011,

Actual value	0	0	0	1	0	0	0	0	1	1
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	1	0	1	1	1	1	0	1	1

Output = 1101111011

For 0001100010,

Actual value	0	0	0	1	1	0	0	0	1	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	1	0	1	0	1	1	0	1	0

Output = 1101011010

Therefore,

XOR of 0001000011 = 1101111011

XOR of 0001100010 = 1101011010

Step 4: Divide both XORed values into two halves and right shift by three.

Given bits are: 1101111011 and 1101011010

Actual bits	1101111011		1101011010	
Before shift	Left half	Right half	Left half	Right half
	11011	11011	11010	11010
After shift	01111	01111	01011	01011

Therefore, after shift

1000011010 = 0111101111

0001001011 = 0101101011

Step 5: Perform inverse E/P (E/P⁻¹) to the output of step 3.

For 0111101111,

E/P⁻¹ table	7	6	2	4	3
Arrangement of bits	1	0	1	1	1

For 0101101011,

E/P⁻¹ table	7	6	2	4	3
Arrangement of bits	1	0	1	1	0

Therefore,

E/P⁻¹ of 1000011010 = 10111

E/P⁻¹ of 0001001011 = 10110

Step 6: Convert binary value into Alphabet by using values from table.

11000 = X

10110 = W

Step 7: Left shift the letters using the set of random numbers.

X left shift by 10 = N

W left shift by 8 = 0

Therefore, Decrypted form of QSIS = NO

4.2. Test two

4.2.1 Encrypting "Hi"

Step 1: Shift N and O by 10 and 8, respectively.

H + 10 = R

I + 8 = Q

Step 2: Convert R and Q into binary by using values from table.

R = 10001

Q = 10000

Step 3: Perform E/P to binary values of X and W

E/p for 10001,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	0	0	1	0	0	0	1	1	1	0

E/P for 10000,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	0	0	0	0	0	0	1	0	1	0

Therefore,

E/P of 10001 = 0010001110

E/P of 10110 = 0000001010

Step 4: Divide the expanded bits into two halves and left shift each half by three.

Given bits are: 0010001110 and 0000001010

Actual bits	0010001110		0000001010	
Before shift	Left half	Right half	Left half	Right half
	00100	01110	00000	01010
After shift	10000	11001	00000	01001

Therefore, after shift,

0010001110 = 1000011001

0000001010 = 0000001001

Step 5: XOR both values with 1 1 0 0 1 1 1 0 0 0

For 1000011001

Actual value	1	0	0	0	0	1	1	0	0	1
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	1	0	0	1	0	0	0	0	1

Output = 0100100001

For 0000001001

Actual value	0	0	0	0	0	0	1	0	0	1
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	1	0	0	1	1	0	0	0	1

Output = 1100110001

Step 6: Perform Permutation 10 with the outputs of Step 5

For 0100100001,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	1	1	0	0	1	0	0	0	0	0

Output = 1100100000

For 1100110001,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	1	1	1	0	1	0	1	0	0	0

Output = 1110101000

Therefore,

P10 of 0100100001 = 1100100000

P10 of 1100110001 = 1110101000

Step 6: Divide the bits into equal parts each of five bits and write the alphabetic value according to the corresponding binary value.

From table,

11001 = Z

00000 = A

11101 = ?

01000 = I

Therefore, encrypted form of “Hi” = ZA?I

4.2.2. Decrypting “ZA?I”

Here, ciphertext = ZA?I

Step 1: Convert cipher text into binary from the table and combine them.

Z = 11001

A = 00000

? = 11101

I = 01000

After combining, cipher text = 11001000001110101000

Step 2: Divide the cipher text into equal parts each of 10-bits and perform inverse permutation 10 ($P10^{-1}$).

First half = 1100100000

Second half = 1110101000

Inverse P10 each half ($P10^{-1}$)

For 1100100000,

$P10^{-1}$ table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	0	1	0	0	1	0	0	0	0	1

For 1110101000,

P10⁻¹ table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	1	1	0	0	1	1	0	0	0	1

Therefore,

P10⁻¹ of 1100100000 = 0100100001

P10⁻¹ of 1110101000 = 1100110001

Step 3: XOR each value with XOR constant (1 1 0 0 1 1 1 0 0 0)

For 0100100001,

Actual value	0	1	0	0	1	0	0	0	0	1
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	0	0	0	0	1	1	0	0	1

Output = 1000011001

For 1100110001,

Actual value	1	1	0	0	1	1	0	0	0	1
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	0	0	0	0	0	1	0	0	1

Output = 0000001001

Therefore,

XOR of 0100100001 = 1000011001

XOR of 1100110001 = 0000001001

Step 4: Divide both XORed values into two halves and right shift by three.

Given bits are: 1000011001 and 0000001001

Actual bits	1000011001		0000001001	
Before shift	Left half	Right half	Left half	Right half
	10000	11001	00000	01001
After shift	00100	01110	00000	01010

Therefore, after shift

1000011001 = 0010001110

0000001001 = 0000001010

Step 5: Perform inverse E/P (E/P^{-1}) to the output of step 3.

For 0010001110,

E/P^{-1} table	7	6	2	4	3
Arrangement of bits	1	0	0	0	1

For 0000001010,

E/P^{-1} table	7	6	2	4	3
Arrangement of bits	1	0	0	0	0

Therefore,

E/P^{-1} of 1000011010 = 10001

E/P^{-1} of 0001001011 = 10000

Step 6: Convert binary value into Alphabet by using values from table.

10001 = R

10000 = Q

Step 7: Left shift the letters using the set of random numbers.

R left shift by 10 = H

Q left shift by 8 = I

Therefore, Decrypted form of ZA?I = HI

4.3. Test three

4.3.1 Encrypting "YES"

Step 1: Shift Y, E and S by 10, 8 and 12, respectively.

Y + 10 = C

E + 8 = M

S + 12 = *

Step 2: Convert C, M and * into binary by using values from table.

C = 00010

M = 01100

* = 11110

Step 3: Perform E/P to binary values of X and W

E/p for 00010,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	0	0	0	1	0	0	0	0	0	1

E/P for 01100,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	1	1	0	0	1	1	0	0	0	0

E/P for 11110,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	1	1	0	1	1	1	1	0	1	1

Therefore,

E/P of 00010 = 0001000001

E/P of 01100 = 1100110000

E/P of 11110 = 1101111011

Step 4: Divide the expanded bits into two halves and left shift each half by three.

Given bits are: 0001000001, 1100110000 and 1101111011

Actual bits	0001000001		1100110000		1101111011	
Before shift	Left half	Right half	Left half	Right half	Left half	Right half
	00010	00001	11001	10000	11011	11011
After shift	10000	01000	01110	00100	11110	11110

Therefore, after shift,

0001000001 = 100001000

1100110000 = 0111000100

1101111011 = 1111011110

Step 5: XOR all values with 1100111000

For 1000001000

Actual value	1	0	0	0	0	0	1	0	0	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	1	0	0	1	1	0	0	0	0

Output = 0100110000

For 0111000100

Actual value	0	1	1	1	0	0	0	1	0	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	0	1	1	1	1	1	1	0	0

Output = 1011111100

For 1111011110

Actual value	1	1	1	1	0	1	1	1	1	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	0	1	1	1	0	0	1	1	0

Output = 0011100110

Step 6: Perform Permutation 10 with the outputs of Step 5

For 0100110000,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	0	1	1	0	1	0	0	0	0	0

Output = 0110100000

For 1011111100,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	0	1	1	1	0	1	1	1	0	1

Output = 0111011101

For 0011100110,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	0	1	0	1	0	1	0	1	1	0

Output = 0101010110

Step 6: Divide the bits into equal parts each of five bits and write the alphabetic value according to the corresponding binary value.

From table,

01101 = N

00000 = A

01110 = O

11101 = ?

01010 = K

10110 = W

Therefore, encrypted form of ‘YES = NAO?KW

4.3.2. Decrypting NAO?KW

Here, ciphertext = NAO?KW

Step 1: Convert cipher text into binary from the table and combine them.

N = 01101

A = 00000

O = 01110

? = 11101

K = 01010

W = 10110

After combining, cipher text = 011010000001110111010101010110

Step 2: Divide the cipher text into equal parts each of 10-bits and perform inverse permutation 10 (P10^{-1}).

First half = 0110100000

Second half = 0111011101

Third half = 0101010110

Inverse P10 each half ($P10^{-1}$)

For 0110100000,

$P10^{-1}$ table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	0	1	0	0	1	1	0	0	0	0

For 0111011101,

$P10^{-1}$ table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	1	0	1	1	1	1	1	1	0	0

For 0101010110,

$P10^{-1}$ table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	0	0	1	1	1	0	0	1	1	0

Therefore,

$P10^{-1}$ of 0110100000 = 0100110000

$P10^{-1}$ of 0111011101 = 1011111100

$P10^{-1}$ of 0101010110 = 0011100110

Step 3: XOR each value with XOR constant (1 1 0 0 1 1 1 0 0 0)

For 0100110000,

Actual value	0	1	0	0	1	1	0	0	0	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	0	0	0	0	0	1	0	0	0

Output = 1000001000

For 1011111100,

Actual value	1	0	1	1	1	1	1	1	0	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	1	1	1	0	0	0	1	0	0

Output = 0111000100

For 0011100110,

Actual value	0	0	1	1	1	0	0	1	1	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	1	1	1	0	1	1	1	1	0

Output = 1111011110

Therefore,

XOR of 0100110000 = 1000001000

XOR of 1011111100 = 0111000100

XOR of 0011100110 = 1111011110

Step 4: Divide both XORed values into two halves and right shift by three.

Given bits are: 1000001000, 0111000100 and 1111011110

Actual bits	1000001000		0111000100		1111011110	
Before shift	Left half	Right half	Left half	Right half	Left half	Right half
	10000	01000	01110	00100	11110	11110
After shift	00010	00001	11001	10000	11011	11011

Therefore, after shift

1000001000 = 0001000001

0111000100 = 1100110000

1111011110 = 1101111011

Step 5: Perform inverse E/P (E/P^{-1}) to the output of step 3.

For 0001000001,

E/P^{-1} table	7	6	2	4	3
Arrangement of bits	0	0	0	1	0

For 1100110000,

E/P^{-1} table	7	6	2	4	3
Arrangement of bits	0	1	1	0	0

For 1101111011,

E/P^{-1} table	7	6	2	4	3
Arrangement of bits	1	1	1	1	0

Therefore,

E/P^{-1} of 0010000010 = 00010

E/P^{-1} of 1001100001 = 01100

E/P^{-1} of 1011110111 = 11110

Step 6: Convert binary value into Alphabet by using values from table.

00010 = C

01100 = M

11110 = *

Step 7: Left shift the letters using the set of random numbers.

C left shift by 10 = Y

M left shift by 8 = E

* left shift by 12 = S

Therefore, Decrypted form of NAO?KW = YES

4.4. Test four

4.4.1. Encrypting ".TXT"

Step 1: Shift ., T, X and T by 10, 8, 12, and 6 respectively.

. + 10 = G

T + 8 = !

X + 12 = D

T + 6 = Z

Step 2: Convert C, M and * into binary by using values from table.

G = 00110

! = 11011

D =00011

Z = 11001

Step 3: Perform E/P to binary values of X and W

E/p for 00110,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	0	1	0	1	1	0	0	0	0	1

E/P for 11011,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	1	0	1	1	0	1	1	1	1	1

E/P for 00011,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	0	0	1	1	0	0	0	1	0	1

E/P for 11001,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	1	0	1	0	0	1	1	1	1	0

Therefore,

E/P of 01011 = 0101100001

E/P of 11011 = 1011011111

E/P of 00011 = 0011000101

E/P of 11001 = 1010011110

Step 4: Divide the expanded bits into two halves and left shift each half by three.

Given bits are: 0101100001, 1011011111, 0011000101 and 1010011110

Actual bits	0101100001		1011011111		0011000101		1010011110	
Before shift	Left half	Right half	Left half	Right half	Left half	Right half	Left half	Right half
	01011	00001	10110	11111	00110	00101	10100	11110
After shift	11010	01000	10101	11111	10001	01001	00101	10111

Therefore, after shift,

0101100001 = 1101001000

1011011111 = 1010111111

0011000101 = 1000101001

1010011110 = 0010110111

Step 5: XOR all values with 1100111000

For 1101001000

Actual value	1	1	0	1	0	0	1	0	0	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	0	0	1	1	1	0	0	0	0

Output = 0001110000

For = 1010111111

Actual value	1	0	1	0	1	1	1	1	1	1
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	1	1	0	0	0	0	1	1	1

Output = 0110000111

For 1000101001

Actual value	1	0	0	0	1	0	1	0	0	1
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	1	0	0	0	1	0	0	0	1

Output = 0100010001

For 0010110111

Actual value	0	0	1	0	1	1	0	1	1	1
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	1	1	0	0	0	1	1	1	1

Output = 1110001111

Therefore,

XOR of 1101001000 = 0001110000

XOR of 1010111111 = 0110000111

XOR of 1000101001 = 0100010001

XOR of 0010110111 = 1110001111

Step 6: Perform Permutation 10 with the outputs of Step 5

For 0001110000,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	0	1	1	0	0	1	0	0	0	0

Output = 0110010000

For 0110000111,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	1	0	0	1	1	0	0	1	1	0

Output = 1001100110

For 0100010001,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	1	0	1	0	1	0	0	0	0	0

Output = 1010100000

For 1110001111,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	1	0	0	1	1	0	1	1	1	1

Output = 1001101111

Therefore,

P10 of 0001110000 = 0110010000

P10 of 0110000111 = 1001100110

P10 of 0100010001 = 1010100000

P10 of 1110001111 = 1001101111

Step 6: Divide the bits into equal parts each of five bits and write the alphabetic value according to the corresponding binary value.

From table,

01100 = M

10000= Q

10011 = T

00110 = G

10101 = V

00000 = A

10011 = T

01111 = P

Therefore, encrypted form of '.TXT' = MQTGVATP

4.4.2. DECRYPTING "MQTGVATP"

Here, ciphertext = MQTGVATP

Step 1: Convert cipher text into binary from the table and combine them.

M = 01100

Q = 10000

T = 10011

G = 00110

V = 10101

A = 00000

T = 10011

01111 = P

After combining, cipher text = 1011100100100110011010101000001001

Step 2: Divide the cipher text into equal parts each of 10-bits and perform inverse permutation 10 ($P10^{-1}$).

First half = 0110010000

Second half = 1001100110

Third half = 1010100000

Fourth half = 1001101111

Inverse P10 each half ($P10^{-1}$)

For 0110010000**,**

$P10^{-1}$ table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	0	0	0	1	1	1	0	0	0	0

For 1001100110,

$P10^{-1}$ table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	0	1	1	0	0	0	0	1	1	1

For 1010100000**,**

$P10^{-1}$ table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	0	1	0	0	0	1	0	0	0	1

For 1001101111**,**

$P10^{-1}$ table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	1	1	1	0	0	0	1	1	1	1

Therefore,

$P10^{-1}$ of 1011100100 = 0001110000

$P10^{-1}$ of 1001100110 = 0110000111

$P10^{-1}$ of 1010100000 = 0100010001

$P10^{-1}$ of 1001101111 = 1110001111

Step 3: XOR each value with XOR constant (1 1 0 0 1 1 1 0 0 0)

For 0001110000,

Actual value	0	0	0	1	1	1	0	0	0	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	1	0	1	0	0	1	0	0	0

Output = 1101001000

For 0110000111,

Actual value	0	1	1	0	0	0	0	1	1	1
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	0	1	0	1	1	1	1	1	1

Output = 1010111111

For 0100010001,

Actual value	0	1	0	0	0	1	0	0	0	1
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	0	0	0	1	0	1	0	0	1

Output = 1000101001

For 1110001111,

Actual value	1	1	1	0	0	0	1	1	1	1
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	0	1	0	1	1	0	1	1	1

Output = 0010110111

Therefore,

XOR of 0001110000 = 1101001000

XOR of 0110000111 = 1010111111

XOR of 0100010001 = 1000101001

XOR of 1110001111 = 0010110111

Step 4: Divide both XORed values into two halves and right shift by three.

Given bits are: 1010101101, 1010111111, 1000101001 and 0010110111

Actual bits	1101001000		1010111111		1000101001		0010110111	
Before shift	Left half	Right half	Left half	Right half	Left half	Right half	Left half	Right half
	11010	01000	10101	11111	10001	01001	00101	10111
After shift	01011	00001	10110	11111	00110	00101	10100	11110

Therefore, after shift

1101001000 = 0101100001

1010111111 = 1011011111

1000101001 = 0011000101

0010110111 = 1010011110

Step 5: Perform inverse E/P (E/P^{-1}) to the output of step 3.

For 0101100001,

E/P^{-1} table	7	6	2	4	3
Arrangement of bits	0	0	1	1	0

For 1011011111,

E/P^{-1} table	7	6	2	4	3
Arrangement of bits	1	1	0	1	1

For 0011000101,

E/P^{-1} table	7	6	2	4	3
Arrangement of bits	0	0	0	1	1

For 1010011110,

E/P^{-1} table	7	6	2	4	3
Arrangement of bits	1	1	0	0	1

Therefore,

E/P^{-1} of 0101100001 = 00110

E/P^{-1} of 1011011111 = 11011

E/P^{-1} of 0011000101 = 00011

E/P^{-1} of 1010011110 = 11001

Step 6: Convert binary value into Alphabet by using values from table.

00110 = G

11011 = !

00011 = D

11001 = Z

Step 7: Left shift the letters using the set of random numbers.

G left shift by 10 = .

! left shift by 8 = T

D left shift by 12 = X

Z left shift by 6 = T

Therefore, Decrypted form of MQTGVATP = .TXT

4.5. Test five

4.5.1. Encrypting "CAT"

Step 1: Shift C, A and T by 10, 8 and 12, respectively.

C + 10 = M

A + 8 = I

T + 12 = £

Step 2: Convert C, M and * into binary by using values from table.

M = 01100

I = 01000

£ = 11111

Step 3: Perform E/P to binary values of X and W

E/p for 01100,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	1	1	0	0	1	1	0	0	0	0

E/P for 01000,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	1	0	0	0	0	1	0	0	0	0

E/P for 11111,

E/P table	2	3	5	4	3	2	1	5	1	4
Arranging bits	1	1	1	1	1	1	1	1	1	1

Therefore,

E/P of 00010 = 1100110000

E/P of 01100 = 1000010000

E/P of 11110 = 1111111111

Step 4: Divide the expanded bits into two halves and left shift each half by three.

Given bits are: 1100110000, 1000010000 and 1111111111

Actual bits	1100110000		1000010000		1111111111	
Before shift	Left half	Right half	Left half	Right half	Left half	Right half
	11001	10000	10000	10000	11111	11111
After shift	01110	00100	00100	00100	11111	11111

Therefore, after shift,

1100110000 = 0111000100

1000010000 = 0010000100

1111111111 = 1111111111

Step 5: XOR all values with 1100111000

For 0111000100

Actual value	0	1	1	1	0	0	0	1	0	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	0	1	1	1	1	1	1	0	0

Output = 1011111100

For 0010000100

Actual value	0	0	1	0	0	0	0	1	0	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	1	1	0	1	1	1	1	0	0

Output = 1110111100

For 1111111111

Actual value	1	1	1	1	1	1	1	1	1	1
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	0	1	1	0	0	0	1	1	1

Output = 0011000111

Therefore,

XOR of 0111000100 = 1011111100

XOR of 0010000100 = 1110111100

XOR of 1111111111 = 0011000111

Step 6: Perform Permutation 10 with the outputs of Step 5

For 1011111100,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	0	1	1	1	0	1	1	1	0	1

Output = 0111011101

For 1110111100,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	0	1	1	1	1	0	1	1	0	1

Output = 0111101101

For 0011000111,

P10 table	10	5	6	3	2	4	1	8	9	7
Arranging bits	1	0	0	1	0	1	0	1	1	0

Output = 1001010110

Therefore,

P10 of 1011111100 = 0111011101

P10 of 1110111100 = 0111101101

P10 of 0011000111 = 1001010110

Step 6: Divide the bits into equal parts each of five bits and write the alphabetic value according to the corresponding binary value.

From table,

01110 = O

11101 = ?

01111 = P

01101 = N

10010 = S

10110 = W

Therefore, encrypted form of CAT = O?PNSW

4.5.2 DECRYPTING "O?PNSW"

Step 1: Convert cipher text into binary from the table and combine them.

O = 01110

? = 11101

P = 01111

N = 01101

S = 10010

W = 10110

After combining, cipher text = 011101110101111011011001010110

Step 2: Divide the cipher text into equal parts each of 10-bits and perform inverse permutation 10 (P10^{-1}).

First half = 0111011101

Second half = 0111101101

Third half = 1001010110

Inverse P10 each half (P10^{-1})

For 0111011101**,**

P10^{-1} table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	1	0	1	1	1	1	1	1	0	0

For 0111101101,

P10^{-1} table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	1	1	1	0	1	1	1	1	0	0

For 1001010110**,**

P10^{-1} table	7	5	4	6	2	3	10	8	9	1
Arrangement of bits	0	0	1	1	0	0	0	1	1	1

Therefore,

P10^{-1} of 0111011101 = 1011111100

P10^{-1} of 0111101101 = 1110111100

P10^{-1} of 1001010110 = 0011000111

Step 3: XOR each value with XOR constant (1 1 0 0 1 1 1 0 0 0)

For 1011111100,

Actual value	1	0	1	1	1	1	1	1	0	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	1	1	1	0	0	0	1	0	0

Output = 0111000100

For 1110111100,

Actual value	1	1	1	0	1	1	1	1	0	0
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	0	0	1	0	0	0	0	1	0	0

Output = 0010000100

For 0011000111,

Actual value	0	0	1	1	0	0	0	1	1	1
XOR constant	1	1	0	0	1	1	1	0	0	0
Output of XOR	1	1	1	1	1	1	1	1	1	1

Output = 1111111111

Therefore,

XOR of 1011111100 = 0111000100

XOR of 1110111100 = 0010000100

XOR of 0011000111 = 1111111111

Step 4: Divide both XORed values into two halves and right shift by three.

Given bits are: 0111000100, 0111000100 and 1111111111

Actual bits	0111000100		0010000100		1111111111	
Before shift	Left half	Right half	Left half	Right half	Left half	Right half
	01110	00100	00100	00100	11111	11111
After shift	11001	10000	10000	10000	11111	11111

Therefore, after shift

0111000100 = 1100110000

0010000100 = 1000010000

1111111111 = 1111111111

Step 5: Perform inverse E/P (E/P^{-1}) to the output of step 3.

For 1100110000,

E/P^{-1} table	7	6	2	4	3
Arrangement of bits	0	1	1	0	0

For 1000010000,

E/P^{-1} table	7	6	2	4	3
Arrangement of bits	0	1	0	0	0

For 1111111111,

E/P^{-1} table	7	6	2	4	3
Arrangement of bits	1	1	1	1	1

Therefore,

E/P^{-1} of 0010000010 = 01100

E/P^{-1} of 1001100001 = 01000

E/P^{-1} of 1011110111 = 11111

Step 6: Convert binary value into Alphabet.

01100 = M

01000 = I

11111 = £

Step 7: Left shift the letters using the set of random numbers.

M left shift by 10 = C

I left shift by 8 = A

£ left shift by 12 = T

Therefore, Decrypted form of O?PNSW = CAT

.

5. Analysis

The cryptographic algorithm created in Task 03, the "Advanced Caesar Cipher," is evaluated in this section. The analysis focuses on the algorithm's complexity, and improved security features to highlight its main advantages. At the same time, it highlights the limitations of that algorithm. This research tries to give a fair assessment of the effectiveness of the algorithm and potential areas for improvement.

5.1. Strength

1. Increased Complexity

   The inclusion of the different processes like conversion into binary, Expansion and Permutation and XOR has increased the complexity of algorithm, making it significantly harder to break.

2. Permutation and Substitution

   The combination of E/P, XOR and permutation have added the extra layer of protection, making the ciphertext less predictable and harder to reverse-engineer.

3. Dynamic Nature

   The use of random numbers for shifts added a dynamic component to the encryption process, which strengthened resistance against frequency analysis.

4. Bitwise Manipulation

   Combining XOR operations with the stage where the expanded bits are divided and left-shifted by three results in a more complex transformation that makes it harder to identify basic patterns. This bitwise operation assures that minor modifications to the plaintext create notable change to the ciphertext.

5. Easy Decryption Process

   If the key and constants are known, the structure of the method allows for simple and reliable decryption. The steps performed in encryption guarantee that every stage of the procedure can reversed during decryption, preserving the original message's integrity.

6. Efficient for Shorter Messages

The inclusion of simple operations like XOR, Shifting and Permutation makes it more efficient for processing short messages. It can easily be implemented on the device that has little processing power.

7. One-to-Two Letter Mapping

In this algorithm, each letter in the plaintext results in two letters in the ciphertext due to the complex transformation process, making it harder to decipher through basic analysis.

5.2. Weaknesses

1. Position-Based Repetition Vulnerability

The encryption of the same letter of different word in the same position produces same result, which makes it vulnerable for frequency analysis or positional attacks, especially for repetitive plaintext.

2. Inability to Encrypt Numbers

This algorithm cannot encrypt the numbers, which limits the versatility of algorithm which could restrict its application in modern systems that manage diverse data types.

3. Repetation of Shift Numbers Weakens Security

The encryption procedure is made predictable by continuously using the same shift number sequence. With sufficient ciphertext observation, an attacker may figure out the pattern and reverse-engineer the shifts.

4. Vulnerability Due to Fixed Encryption Parameters

All the encryption and decryption steps depend on a set of fixed elements, including shifting letters by a set of random numbers, E/p table, P10 table, XOR constant, inverse E/P table, and inverse P10 table. Since these elements remain the same for all encryptions and decryptions, their exposure would compromise the security of the entire algorithm.

5. Potential for Human Error

Both encryption and decryption have a substantial risk of human error because of the many processes (such as shifts, bit manipulations, and table lookups). An error in implementing any step while performing encryption or decryption could result in data loss or inaccurate output if any stage is implemented incorrectly.

6. May not be able to process the letters that are represented by more than 5-bits.

This algorithm is not applicable for letters that are represented by more than 5-bits. This is due to the XOR constant being ten bits, while both the P10 and E/P tables are designed to work with 5-bit blocks, as seen in the binary representations of characters. As the algorithm is unable to process binary values larger than 5 bits, any input exceeding this size may result in data loss. The same limitation applies to the inverse P10 and inverse E/P tables, while decrypting which are also restricted to 5-bit blocks.

5.3. Application Areas

1. Personal Security: The newly developed algorithm can be used for personal security by encrypting personal files and folders stored on personal devices.
2. Understanding Algorithm Complexity: It can be used to demonstrate how the minor changes like adding random keys or using bitwise operations, significantly increases algorithm security and complexity.
3. Lightweight Data Encryption: The algorithm can also be used to encrypt non-critical data such as configuration files, logs, or communication within a controlled environment.
4. Internal Messaging: It can be used to encrypt messages between devices or users to a layer of privacy in local communication.
5. Low-Resource Scenario: The developed algorithm can be ideal for the systems with limited computational resources, such as small, embedded devices or old hardware devices.

6. Conclusion

In today’s world, information has become one of the most important assets and keeping it safe is very necessary. This project explains the basics of IT security and cryptography, focusing on how they help maintain Confidentiality, Integrity, and Availability (CIA) of information. It explores the several types and methods of IT security, making it easier to understand how organizations can protect their valuable data. It also explains the history of cryptography, from ancient times to modern techniques, and shows how we can balance simplicity and security while designing cryptographic systems.

The Caesar cipher, which is an old and simple method of encryption, was chosen as the foundation for this study. While the original Caesar cipher is easy to use and great for learning purposes, it has many weaknesses, like being easily cracked through frequency analysis and having a static key. To overcome these problems, this project introduced an advanced version of the Caesar cipher, adding more complexity to make it secure.

The improvements in the new cipher include generating dynamic keys, converting letters into binary, expanding and permutating bits, and using XOR operations. All these steps make the relationship between the original text and the encrypted text very unclear, making it difficult for hackers to break. Additional steps like shifting bits and applying multiple permutations added even more complexity, making the encryption stronger and more unpredictable.

The modified algorithm was evaluated with different inputs, and the results showed that it is strong enough to protect data. The testing proved its ability to maintain confidentiality and resist attacks like reverse-engineering. However, it was also noted that the new algorithm is more complex and takes more time to process. These findings teach us that while creating strong encryption is important, it is also necessary to keep it practical for real-world use.

This project shows how important it is to keep improving cybersecurity methods. By learning from the weaknesses of older systems and adding new techniques, this coursework combines theory with practical application. It also highlights that cybersecurity is always evolving, and we need to keep updating our methods to stay ahead of threats.

References

agoffe, 2021. *What is IT Security?.* [Online]
Available at: https://www.crowdstrike.com/en-us/cybersecurity-101/cybersecurity/it-security/
[Accessed 12 December 2024].

Burge, S., 2023. *What is Physical Security & Why is it Needed?.* [Online]
Available at: https://internationalsecurityjournal.com/what-is-physical-security/
[Accessed 12 December 2024].

cloudflare, 2024. *What is network security?.* [Online]
Available at: https://www.cloudflare.com/learning/network-layer/network-security/
[Accessed 12 December 2024].

cryptomathis, 2024. *Symmetric Key Encryption - why, where and how it's used in banking.* [Online]
Available at: https://www.cryptomathic.com/blog/symmetric-key-encryption-why-where-and-how-its-used-in-banking
[Accessed 12 December 2024].

Fasulo, P., 2024. *What is the CIA Triad? Definition, Importance, & Examples.* [Online]
Available at: https://securityscorecard.com/blog/what-is-the-cia-triad/
[Accessed 3 January 2025].

forcepoint, 2024. *What is Network Security?.* [Online]
Available at: https://www.forcepoint.com/cyber-edu/network-security
[Accessed 12 December 2024].

Fortinet , 2024. *CIA Triad.* [Online]
Available at: https://www.fortinet.com/resources/cyberglossary/cia-triad
[Accessed 9 December 2024].

fortinet, 2024. *Operational Security (OPSEC).* [Online]
Available at: https://www.fortinet.com/resources/cyberglossary/operational-security
[Accessed 12 December 2024].

Jackob, M., 2021. *History of Encryption,* Bethesda: SANS Institute.

javatpoint, 2024. *Caesar Cipher Technique.* [Online]
Available at: https://www.javatpoint.com/caesar-cipher-technique
[Accessed 9 December 2024].

Jim Holdsworth, M. K., 2024. *What is information security (InfoSec)?.* [Online]
Available at: https://www.ibm.com/topics/information-security
[Accessed 12 December 2024].

Michael E. Whitman, H. J. M., 2012. *PRINCIPLES OF INFORMATION SECURITY.* 4th ed. Boston, MA: Cengage Learning.

Microsoft, 2024. *What is information security (InfoSec)?.* [Online]
Available at: https://www.microsoft.com/en-us/security/business/security-101/what-is-information-security-infosec?msockid=24229bd406396f4536da8e9e07bc6edd
[Accessed 12 December 2024].

Osazuwa, O. M. C., 2023. Confidentiality, Integrity, and Availability in Network Systems: A Review of Related Literature. *International Journal of Innovative Science and Research Technology,* 8(12), p. 10.

S.Suguna, D. M., 2016. A Study on Symmetric and Asymmetric Key Encryption Algorithms. *International Research Journal of Engineering and Technology (IRJET),* 03(04), p. 5.

Satish Dadasaheb Tribhuvan, S. A. S., 2024. International Research Journal of Modernization in Engineering Technology and Science. *CAESAR CIPHER TECHNIQUES,* 6(3), p. 4.

splunk, 2024. *The Caesar Cipher, Explained.* [Online]
Available at: https://www.splunk.com/en_us/blog/learn/caesar-cipher.html
[Accessed 9 December 2024].

Teach Computer Science, 2024. *Symmetric Encryption.* [Online]
Available at: https://teachcomputerscience.com/symmetric-encryption/
[Accessed 12 December 2024].