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: Evani Raut

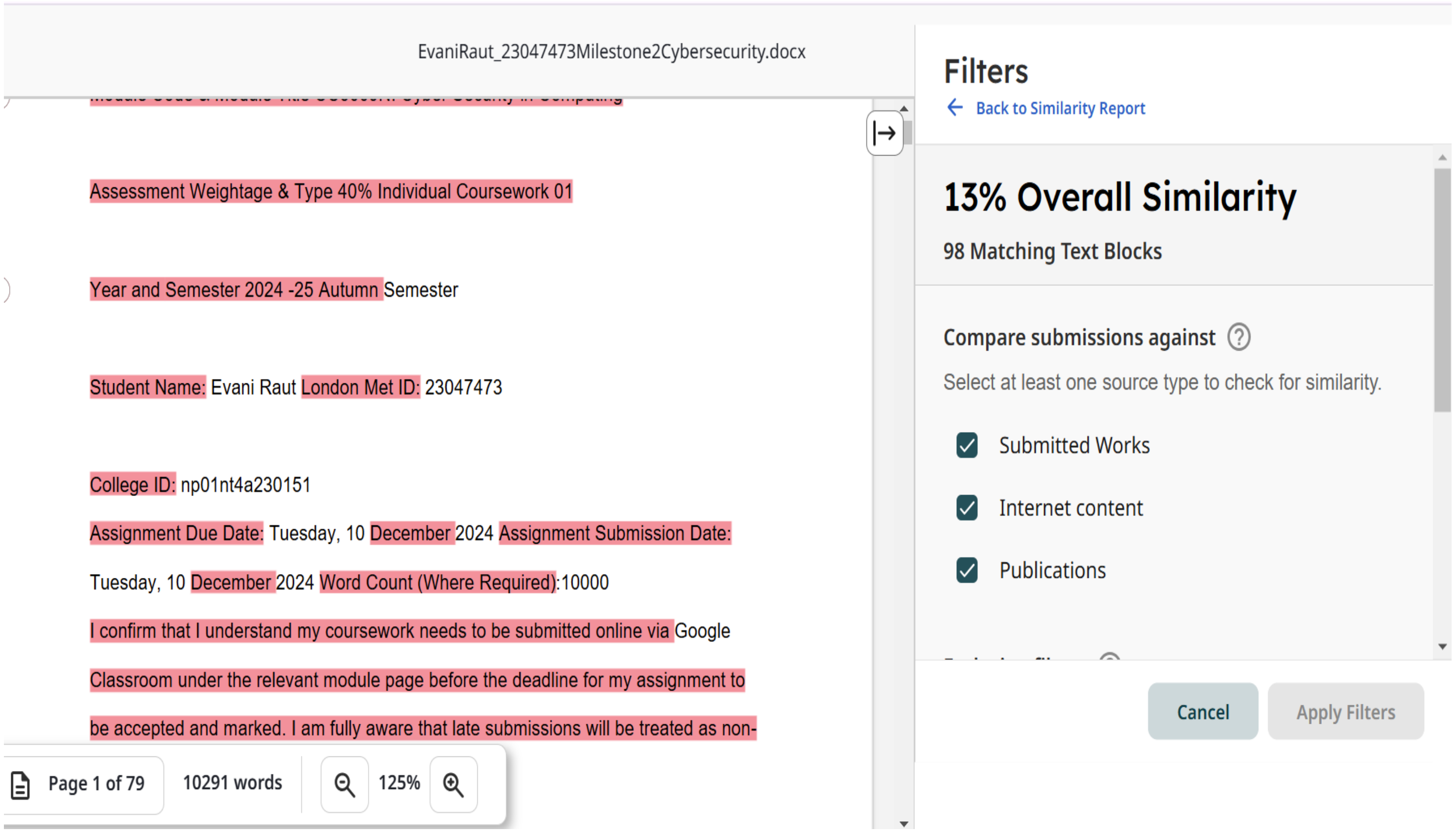
London Met ID: 23047473

College ID: np01nt4a230151

Assignment Due Date: Monday, 20 January 2025

Assignment Submission Date: Monday, 20 January 2025

Word Count (Where Required):10000



EvaniRaut_23047473Milestone2Cybersecurity.docx

Assessment Weightage & Type 40% Individual Coursework 01

Year and Semester 2024 -25 Autumn Semester

Student Name: Evani Raut London Met ID: 23047473

College ID: np01nt4a230151

Assignment Due Date: Tuesday, 10 December 2024 Assignment Submission Date: Tuesday, 10 December 2024 Word Count (Where Required):10000

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-

Page 1 of 79 10291 words 125%

Filters

Back to Similarity Report

13% Overall Similarity

98 Matching Text Blocks

Compare submissions against

Select at least one source type to check for similarity.

- [✓] Submitted Works
- [✓] Internet content
- [✓] Publications

Cancel Apply Filters

Acknowledgment

I want to take a moment to sincerely thank everyone who supported me throughout the completion of this coursework. First, I am deeply grateful to my subject teacher for his insightful guidance, feedback, and constant encouragement. His advice has been valuable in shaping the direction and quality of my coursework.

I am so thankful to Islington College and London Metropolitan University for providing access to the resources and a supportive learning environment that made this research possible.

A want to say a big thank you to my friends and my subject teacher for always being there for me and cheering me on during this journey. Their trust and encouragement gave me confidence to stay focused and overcome challenges along the way.

Lastly, I also want to acknowledge the incredible researchers and authors whose work I relied on for this coursework. Their insights and contributions to the field of biometrics have been an incredible source of knowledge and inspiration for my study.

Thank you all for being part of this journey with me. Your support has meant alot.

Abstract

As digital communication and data exchange continue to grow, the risks associated with cyber threats grows too. This report explores the development of the Inverted Key Cipher (IKC), a new cryptographic algorithm designed to improve the security of sensitive information. The motivation behind this project is to address the growing concerns over the effectiveness of traditional encryption methods in protecting against modern attacks.

The key problem tackled by this work is the increasing vulnerability of current encryption techniques, which are susceptible to methods like brute-force attacks and statistical analysis. The goal was to create a stronger encryption method that could offer better protection while remaining versatile for various applications. This report walks through the process of designing, implementing, and testing the IKC algorithm, while also evaluating its potential real-world uses. The project involved developing the core elements of the IKC, testing its performance, and assessing how well it could be applied in different scenarios. The results show that the IKC provides a higher level of complexity and flexibility than many existing methods. However, it also brings about challenges such as higher computational requirements and difficulties in managing keys. This research holds considerable importance for areas like secure messaging, file encryption, and IoT security, where strong encryption is essential. While the IKC shows promise as an innovative approach to cryptography, further refinement is needed to fully address its limitations. Ultimately, this project contributes to the ongoing development of information security by offering fresh insights into how cryptographic algorithms can evolve to meet new challenges.

Table of Contents

Table of Figures:

Table of Tables:

1. Introduction:

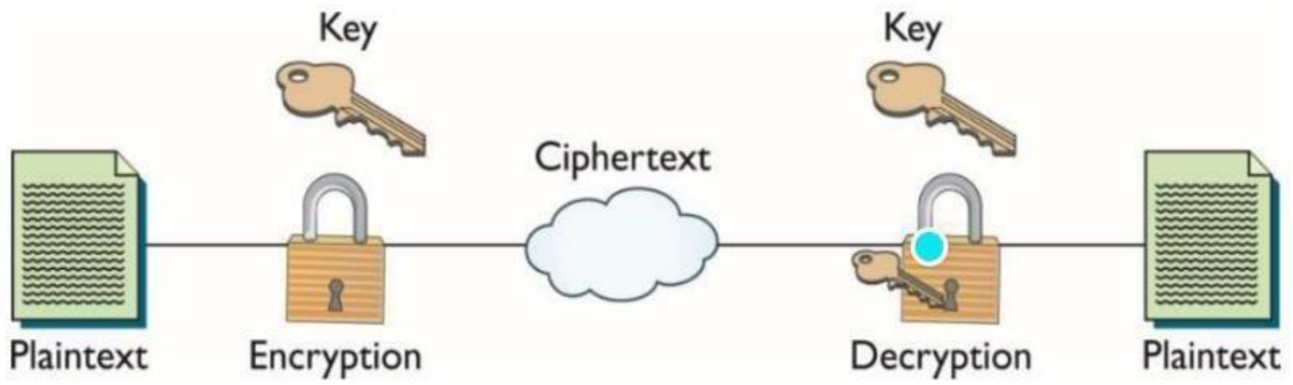


Figure 1:Cryptography

Cryptography is the method of protecting information and communication by encoding it, ensuring that only those with proper authorization can access it. This prevents unauthorized individuals from gaining access to sensitive data. The term "crypt" originates from the Greek word for "hidden," while "graphy" means "writing." In other words, cryptography shields communication by making it unreadable to anyone who isn't authorized. This ancient practice continues to play a critical role in protecting sensitive data, such as those involved in banking, password management, and online transactions (geeksforgeeks, 2024).

1.1 Methods:

Ancient Methods

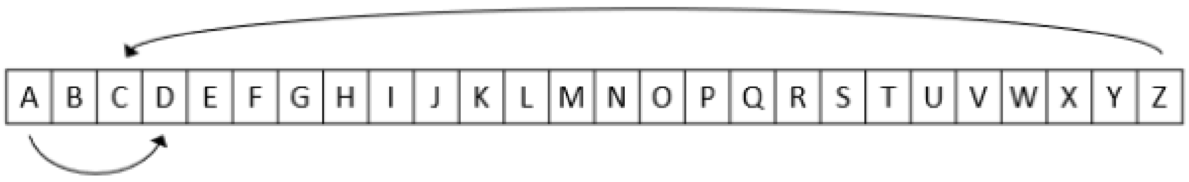


Figure 2: Ancient Method(Ceasar Cipher)

Caesar Cipher: This classic encryption technique, named after Julius Caesar, shifts the letters of the plaintext by a set number of places in the alphabet. While it's simple and easy to apply, it is vulnerable to brute force attacks and frequency analysis (geeksforgeeks, 2024).

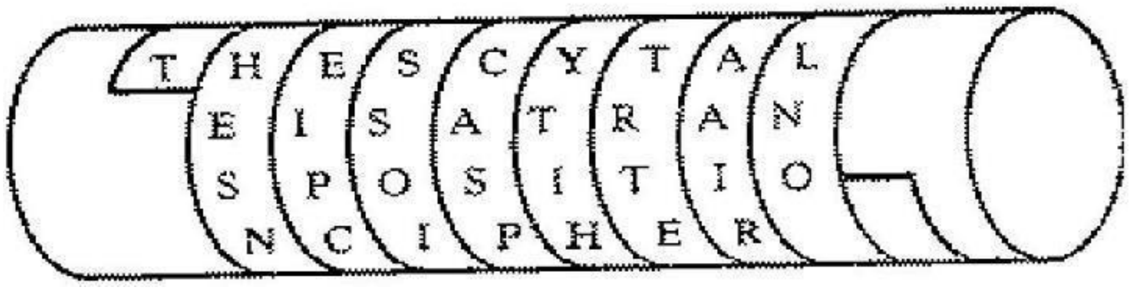


Figure 3:Ancient Method(Scytale Cipher)

Scytale Cipher: A transposition cipher from ancient Sparta, the Scytale cipher involves wrapping parchment around a cylinder and writing across it. To decrypt the message, the recipient needs an identical cylinder to align the characters correctly (dencode, 2024).

Medieval Cryptography

Figure 4:Medieval Cryptography

Medieval cryptography evolved from basic substitution methods to more complex systems, reflecting the era's political and military demands and became more advanced with the introduction of substitution and transposition ciphers. A notable example is the **Vigenère Cipher** which utilized polyalphabetic substitution to enhance security, making it more difficult to decryption than simpler methods like the Caesar Cipher (Nickpelling, 2008).

Toward the late medieval period, cryptographic methods evolved further with the introduction of polyalphabetic ciphers, as seen in Leon Battista Alberti's cipher wheel in 1467. This tool employed polyalphabetic encryption, utilizing multiple shifting alphabets to counteract frequency analysis. Simultaneously, cryptographers like Johannes Trithemius made strides in steganography, the practice of concealing messages within ordinary objects or texts. His seminal work, *Polygraphia*, combined elements of encryption and hidden messaging, demonstrating the sophisticated cryptographic methods of the Renaissance. These developments reflect the increasing complexity and critical role of cryptography in medieval period (Mariana, 2024).

Modern Cryptography

The 20th century saw the rise of machine-based cryptography, which brought major advancements in secure communication. One of the prime example was the Enigma Machine, which was used extensively by Nazi Germany during World War II. The successful cracking of Enigma by Allied codebreakers marked a milestone in the progress of cryptographic science.

Today, modern cryptography relies on highly sophisticated algorithms and encryption methods, with the Advanced Encryption Standard (AES) being one of the most widely used. AES, especially with 128-bit and 256-bit keys, is regarded as one of the most secure encryption methods, providing strong protection that is nearly unbreakable with current computational power. Long encryption keys, such as 128-bit and 256-bit, make encrypted data protected against brute-force attacks. These cryptographic innovations are essential for protecting sensitive information, from online transactions to secure communications in today's digital world (cpl.thalesgroup, 2024).

1.2 Aims and Objectives

Aims

The main goal of this project is to investigate the field of cryptography and information security, with a focus on the development, analysis, and improvement of cryptographic algorithms. The project involves understanding foundational security concepts by selecting a cryptographic algorithm, modifying it to improve its security, and ultimately creating a new, secure encryption and decryption system. This will provide insights into how cryptographic algorithms can evolve to address modern security challenges.

Objectives

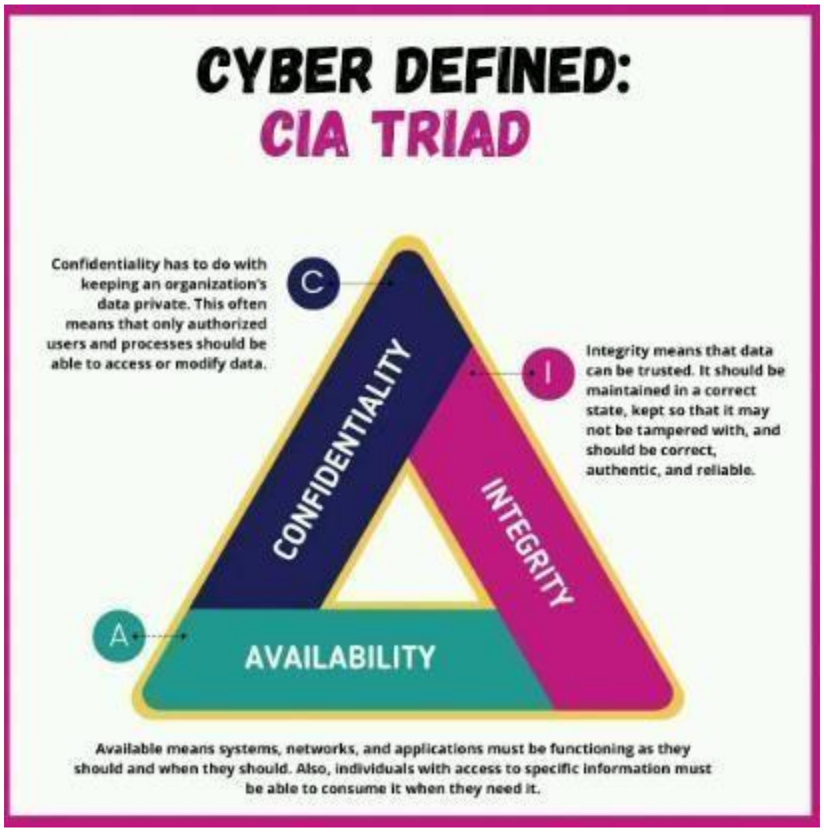
1. To explore and analyze cryptography by introducing its history, key terminologies, and applications, as well as investigating the differences between symmetric and asymmetric encryption systems.
2. To select and analyze an existing cryptographic algorithm, thoroughly explaining its background, and demonstrating encryption and decryption processes with examples.
3. To evaluate the strengths and weaknesses of the selected cryptographic algorithm, considering its practical use cases and performance in securing communications.
4. To research potential improvements to the selected cryptographic algorithm by examining possible mathematical or logical modifications that can enhance its security and efficiency.
5. To design and create a new cryptographic algorithm, applying the modifications and insights gained from the research to develop a more secure encryption and decryption method.
6. To critically analyze the newly created algorithm by explaining the rationale behind the modifications, outlining the encryption and decryption processes, and assessing its strengths and weaknesses.
7. To provide practical examples of the new algorithm in action, showcasing how it encrypts and decrypts data securely, and offering test cases that demonstrate its functionality.
8. To identify the potential applications of the new cryptographic algorithm and discuss how it could be used in real-world security scenarios.

1.3 Fundamentals of Security

Security

Security is essential for protecting computer systems and data from a range of threats, such as theft, damage, and unauthorized access. Cybercriminals often take advantage of weak security measures in vulnerable systems, highlighting the importance of securing information. Many users are attacked because their security measures aren't strong enough to prevent intruders, and criminals are quick to exploit these weaknesses. Strong computer security ensures the privacy, availability, and integrity of both systems and data, protecting them from threats, whether they come from inside or outside the organization and whether they are intentional, like hacking, or accidental (Ron Ross, 2017).

1.4 The CIA Triad (Confidentiality, Integrity, Availability)



Confidentiality has to do with keeping an organization's data private. This often means that only authorized users and processes should be able to access or modify data.

Integrity means that data can be trusted. It should be maintained in a correct state, kept so that it may not be tampered with, and should be correct, authentic, and reliable.

Available means systems, networks, and applications must be functioning as they should and when they should. Also, individuals with access to specific information must be able to consume it when they need it.

Figure 5:CIA Triad

The CIA Triad is a fundamental idea in cybersecurity, representing three key principles that are vital for safeguarding sensitive information and ensuring secure systems. These principles help organizations protect their data and maintain trust with both users and stakeholders.

Confidentiality guarantees that sensitive data is only accessible to authorized individuals, protecting it from unauthorized access and privacy violations. It plays a vital role in securing personal, financial, and business information, ensuring that only those with the right permissions can view or handle it. Techniques like encryption and secure data transmission are commonly used to maintain confidentiality (SentinelOne, 2024).

Integrity guarantees that data is trustworthy, complete, and free from unauthorized modifications. It ensures that information remains reliable and consistent over time.This principle is especially important in industries like healthcare and finance, where data integrity is fundamental to ensuring both security and accuracy (informationsecurity, 2024).

Availability refers to the data being accessible when it is required. The goal is to ensure that critical data and services are always accessible to authorized users when necessary. It becomes even more important when there’s a risk of disruption, such as during denial-of-service (DoS) attacks or system failures. Ensuring availability guarantees that business operations can continue smoothly and without unnecessary interruption (Josh Lake, 2023).

The Role of CIA in Information Security

Figure 6:CIA in Information Security

The **CIA Triad** plays a significant role in strengthening data security by shaping essential practices and policies within organizations. Role-Based Access Control (RBAC) is a crucial practice influenced by the CIA Triad, ensuring that only authorized individuals, based on their specific roles, have access to sensitive data. This significantly lowers the risk of unauthorized access and potential data breaches (Anthony Henderson, 2024).

Encryption is a essential component of the CIA Triad, ensuring data protection by converting it into unreadable formats. This means that even if unauthorized access occurs, the data remains secure and cannot be understood without the proper decryption key, preserving both confidentiality and integrity. By transforming sensitive information into an inaccessible form, encryption plays a crucial role in safeguarding data, minimizing the risk of unauthorized access, and ensuring that it stays protected even in the event of a breach.

Hashing is a technique that transforms data into a fixed-size string, enabling verification without revealing the original information. It is commonly used to check the integrity of files, ensuring they haven't been changed or tampered with during storage or transmission (Margaret Rouse, 2022).

Self-healing backups and replicas ensure the availability of critical data by maintaining secure copies that can be restored in case of data corruption, loss, or other issues. These methods are important for maintaining business continuity and ensuring that information is always accessible when needed.

By implementing these practices, businesses can reinforce their security posture, improve compliance with security standards, and ensure that critical data remains protected, accurate, and available at all times.

1.5 History of Cryptography

Figure 7:History of Cryptography

Cryptography has a rich history, evolving from basic secret writing techniques to advanced algorithms that protect digital information today. The first known use of cryptography dates back to around 1900 BC, with encrypted messages found in ancient Egyptian tombs. The use of cryptography spread throughout the ancient world, with the Caesar cipher, which Julius Caesar used around 100 BC, being one of the most famous early examples. This substitution cipher shifted letters in the alphabet to protect military messages (Gustavus J. Simmons, 2024).

In the medieval and Renaissance periods, cryptography became more advanced. A major development was the **Vigenère cipher**, created in the 16th century. This cipher used a keyword to determine the shifting of letters in a message, making it significantly more complex than previous ciphers. It was considered nearly unbreakable for centuries until advancements in cryptanalysis allowed it to be deciphered (bundesdruckerei, 2022).

The 20th century brought about the use of machines in cryptography. One of the most famous examples is the Enigma machine, utilized by Nazi Germany in World War II. The machine's complex system of rotors generated ciphers that were thought to be impenetrable. However, a team of British cryptanalysts, led by Alan Turing, famously broke the Enigma code, contributing greatly to the Allied victory (thalesgroup, 2023).

Cryptography entered a new phase with the rise of computers. In the 1970s, the Data Encryption Standard (DES) was created and became the first widely accepted method for secure communication. But by the 1990s, increasing computing power uncovered flaws in DES, leading to the development of the more secure Advanced Encryption Standard (AES). Additionally, the development of asymmetric encryption methods, such as the RSA algorithm, enabled secure communication using two keys one for encryption and another for decryption.

Today, cryptography is an integral part of modern technology, securing everything from internet transactions to encrypted messaging apps. It continues to evolve, with new techniques such as **Elliptic Curve Cryptography (ECC)** providing even more efficient and secure solutions

1.6 Cryptosystems

About Cryptosystem

A cryptosystem is a structured designed to ensure the security of data through the use of cryptographic algorithms and protocols. It includes processes for encryption, decryption, and key management to ensure the confidentiality, integrity, and authenticity of the information being transmitted. Cryptosystems are important for securing communications and transactions in a digital environment, protecting information from unauthorized access and alteration.

Modern cryptosystems, which incorporate both symmetric and asymmetric encryption techniques, are used in various applications such as secure communications, online banking, and digital signatures. As cryptosystems evolve, they must address new challenges, such as the potential threat posed by quantum computing, which could compromise many current encryption methods (inviul, 2016).

1.7 Types of Cryptosystems

1.Symmetric Encryption

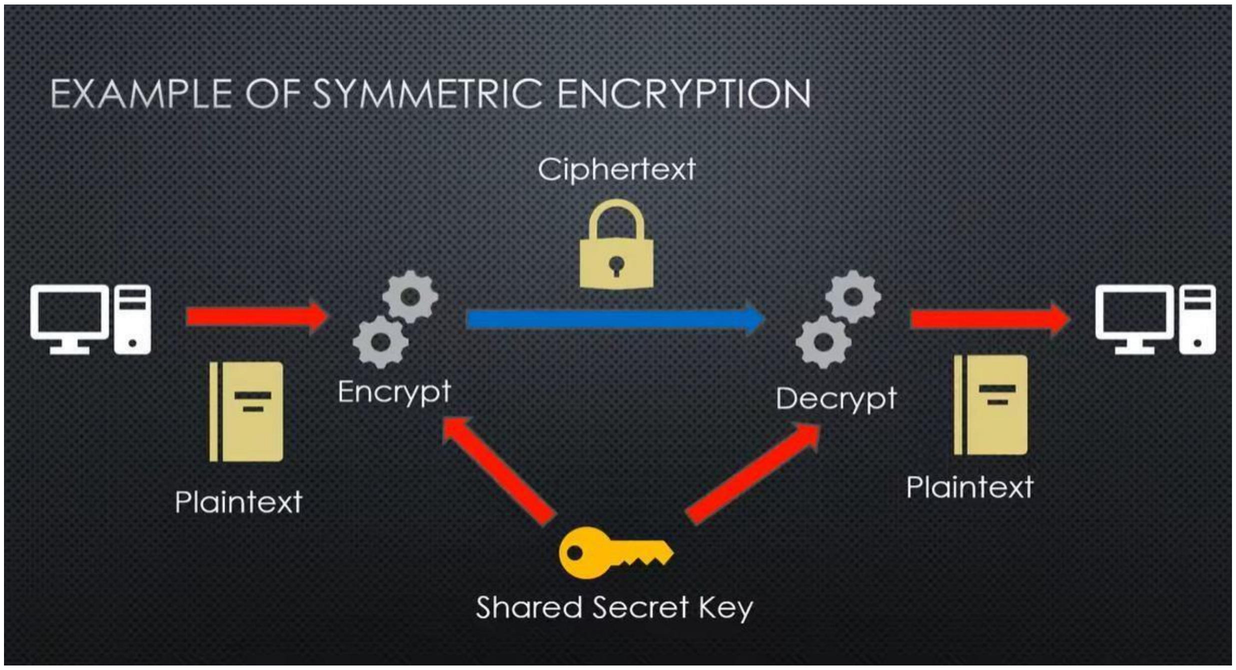


Figure 8:Symmetric Encryption

Definition: Symmetric encryption uses a single key for both the encryption and decryption of data which makes them highly efficient for encrypting large amounts of data, as the algorithms are typically faster than those used in asymmetric systems. This key must remain private and securely shared between the communicating parties to ensure confidentiality.

Example Algorithm:

Advanced Encryption Standard (AES): AES works on fixed-size data blocks of 128 bits and supports key lengths of 128, 192, or 256 bits. It combines substitution and permutation techniques to convert plaintext into ciphertext.

Strengths:

Symmetric encryption is highly efficient, as it is faster compared to asymmetric encryption, making it well-suited for encrypting large volumes of data quickly and effectively. Its straightforward approach relies on the use of a single shared key for both encryption and decryption, simplifying the overall process. This simplicity contributes to its widespread adoption in scenarios where speed and efficiency are critical, such as securing large datasets or real-time communication.

Weaknesses:

Symmetric encryption faces challenges in securely sharing the key between parties. If the key is intercepted, the entire system is compromised, creating a significant vulnerability. Additionally, symmetric encryption struggles with scalability. As the number of users increases, the number of unique keys required grows exponentially, making it less practical for large networks (argoox, 2024).

2 Asymmetric Encryption

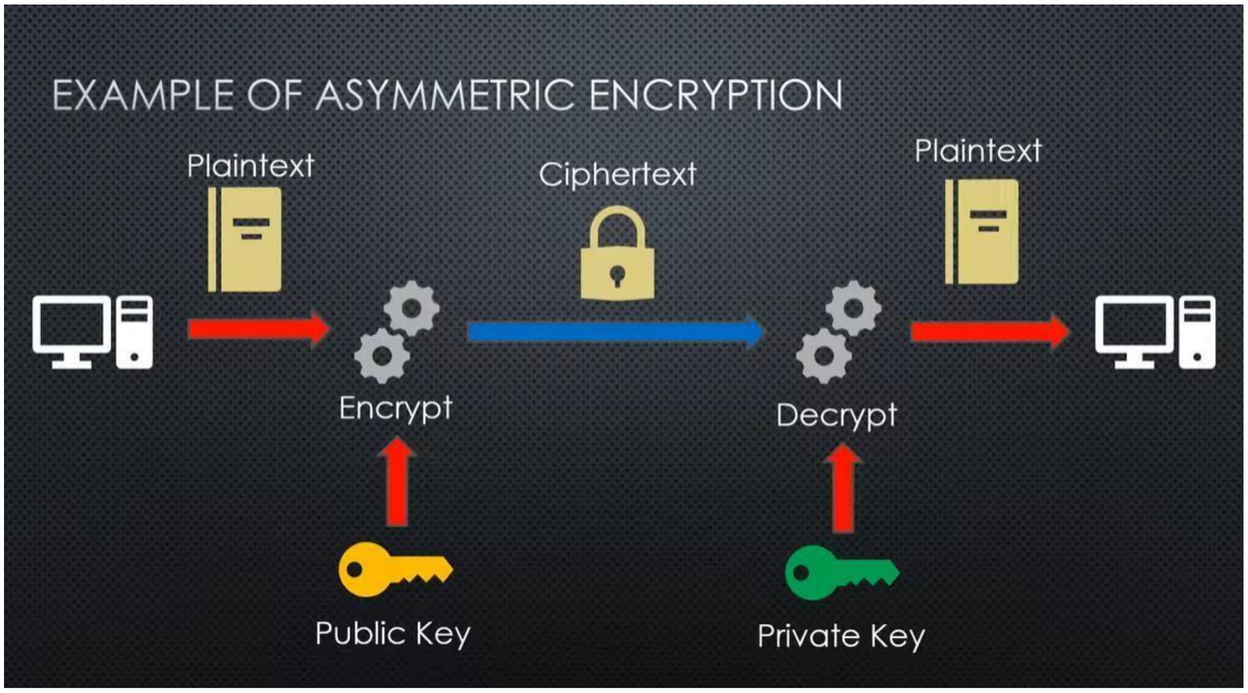


Figure 9:Asymmetric Encryption

Definition: Asymmetric encryption uses a pair of keys, a public key for encryption and a private key for decryption. The public key is shared openly, while the private key is kept secret by its owner.

Example Algorithms:

- **Rivest-Shamir-Adleman (RSA)**: Based on the difficulty of factoring large integers, RSA is widely used for secure key exchanges and digital signatures.
- **Elliptic Curve Cryptography (ECC)**: Leverages the mathematics of elliptic curves to provide similar security to RSA but with smaller key sizes, making it efficient for devices with limited computational power.

Strengths:

Asymmetric encryption addresses the key distribution problem by requiring only the public key to be shared, eliminating the need for secure key exchange between parties.

This simplifies the process of communication, as the public key can be openly distributed while the private key remains confidential. Additionally, asymmetric encryption provides a high level of security, making it particularly effective for authentication and digital signatures. The use of a public and private key pair ensures that data remains secure and verifiable, even in open networks, offering strong protection against unauthorized access.

Weaknesses:

Asymmetric encryption is slower than symmetric encryption due to its complex mathematical operations, which involve key pairs for encryption and decryption. These operations require more computational power, making asymmetric encryption less efficient for encrypting large amounts of data. Additionally, the increased resource intensity can be a limitation in resource-constrained environments, such as devices with limited processing capabilities or in scenarios where efficiency is crucial (allcryptowhitepapers, 2024).

The primary distinction between Symmetric and Asymmetric key cryptography lies in the keys used for encryption and decryption. In Symmetric key cryptography, the same key is used for both encrypting and decrypting data, meaning decryption is only possible if the encryption key is available. On the other hand, Asymmetric key cryptography uses a pair of keys: a public key for encryption and a private key for decryption. Symmetric cryptography employs algorithms such as Stream and Block ciphers.

2. Caesar Cipher

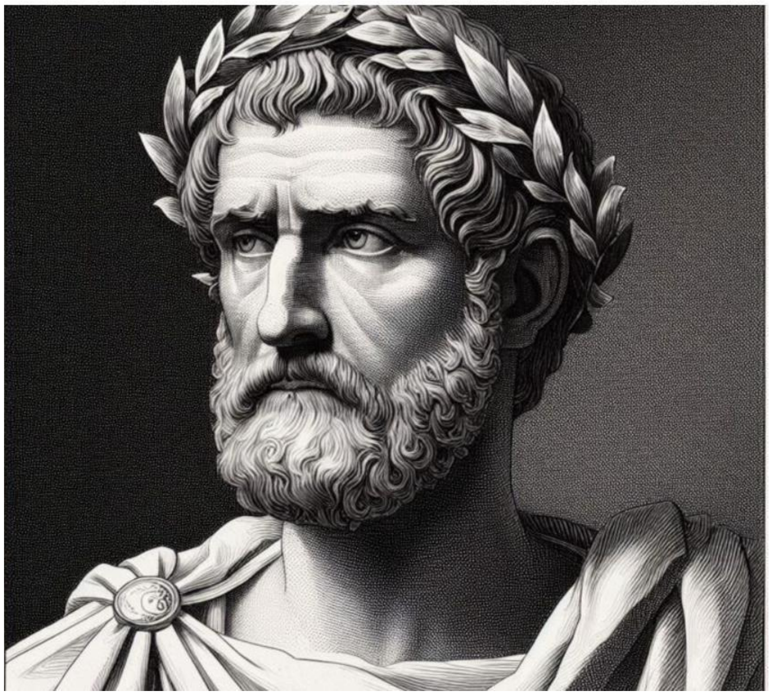


Figure 10:Caesar Cipher

The Caesar cipher is one of the simplest and most renowned encryption methods which was named after Julius Caesar, who used it to secure his private communications with his military commanders. This technique relies on shifting the alphabet's letters by a specific number of positions. While the Caesar cipher is easy to understand and implement, it is highly vulnerable to decryption through brute force or frequency analysis due to the limited number of possible shifts only 25 options for the English alphabet. Despite its simplicity and historical significance, the Caesar Cipher provides an excellent foundation for understanding the basic principles of encryption and decryption. It offers a clear and intuitive approach to cryptography, making it an ideal starting point for further exploration and development (Ofir Kuperman, 2022).

The **Caesar Cipher** is a **substitution cipher** where each letter in the plaintext is shifted a fixed number of positions down or up the alphabet. This shift is determined by a secret key (the number of positions) that both the sender and the receiver must know in advance. The algorithm is symmetric, meaning the same key is used for both encryption and decryption.

Encryption: Each letter in the plaintext is replaced by another letter at a fixed position in the alphabet, determined by the key.

Decryption: The receiver shifts the letters of the ciphertext in the opposite direction by the same number of positions to retrieve the original plaintext.

2.1 The History and Evolution of the Caesar Cipher

Figure 11:The History and Evolution of the Caesar Cipher

The history of the Caesar cipher dates back to ancient Rome, where it was employed by Julius Caesar to secure his military communications. According to historical sources, Caesar used this encryption technique as early as 58 BCE during his military campaigns in Gaul. The cipher involved substituting each letter of the alphabet with another letter located a fixed number of positions away. Caesar typically used a shift of three positions, which ensured that only those who knew the key could decipher the messages (dlab.epfl, 2024).

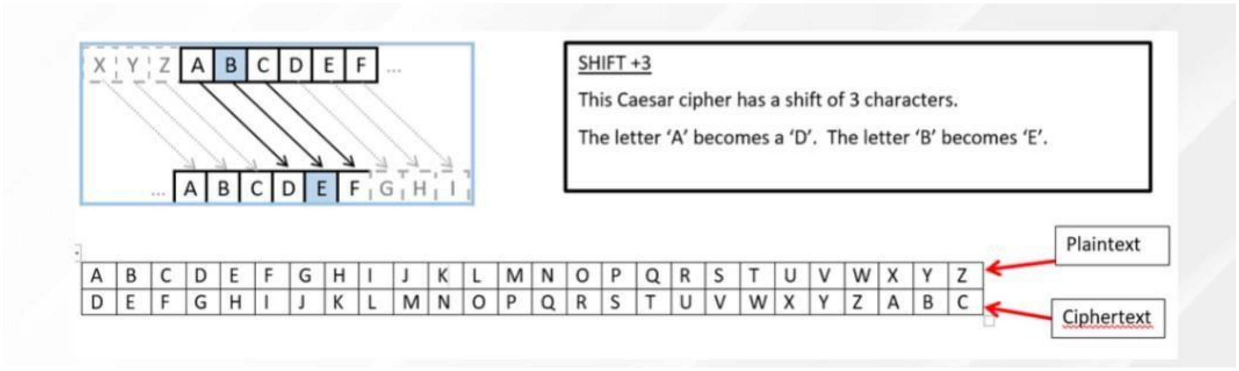
This cipher was significant in an era when literacy was rare and code-breaking techniques were virtually nonexistent. Caesar’s use of this method highlights its effectiveness as a simple yet reliable means of securing information. Historical records, such as those by the Roman historian Suetonius, document its use in Caesar’s personal and official correspondence.

Later, the Caesar cipher inspired more sophisticated cryptographic systems. The technique's simplicity made it easy to replicate and adapt. For example, Augustus, Caesar’s successor, reportedly employed a similar cipher but with a shift of one position, further showing its practicality for varying levels of security

2.2 Reasons for Selecting Caesar Cipher:

The Caesar Cipher is selected for its simplicity, as it is easy to understand and implement. It provides a straightforward introduction to the concept of substitution ciphers, which are fundamental to the study of cryptography. Additionally, its historical significance cannot be overlooked, as it is one of the earliest recorded encryption techniques, originally used by Julius Caesar to safeguard his communications. This adds a rich historical context to its study. Furthermore, the Caesar Cipher offers substantial educational value. By examining this cipher, I can explore how modern cryptographic methods have evolved from such basic techniques, making it a perfect starting point for delving into more complex encryption algorithms.

2.3 How it works:


X Y Z A B C D E F ...
... A B C D E F G H I
SHIFT +3
This Caesar cipher has a shift of 3 characters.
The letter ‘A’ becomes a ‘D’. The letter ‘B’ becomes ‘E’.
Plaintext
Ciphertext

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

Figure 12: Working Mechanism

In the Caesar cipher, each letter of the plaintext is substituted with a letter that is a fixed number of positions down or up the alphabet. For example, with a shift of 3, 'A' becomes 'D', 'B' becomes 'E', 'C' becomes 'F', and so on. After reaching 'Z', the alphabet wraps around, so 'Z' becomes 'C'.

Example:

For Letter: E

Position of E in the alphabet: 5

Add the shift value: 5+3=85 + 3 = 85+3=8

The letter at position 8 is H.

Result so far: H

For Letter: V

Position of V in the alphabet: 22

Add the shift value: 22+3=2522 + 3 = 2522+3=25

The letter at position 25 is Y.

Result so far: HY

For Letter: A

Position of A in the alphabet: 1

Add the shift value: 1+3=41 + 3 = 41+3=4

The letter at position 4 is D.

Result so far: HYD

For Letter: N

Position of N in the alphabet: 14

Add the shift value: 14+3=1714 + 3 = 1714+3=17

The letter at position 17 is Q.

Result so far: HYDQ

For Letter: I

Position of I in the alphabet: 9

Add the shift value: 9+3=129 + 3 = 129+3=12

The letter at position 12 is L.

Final Encoded Text: HYDQL

Step 3: Decoding the Cipher

Decoding involves reversing the process by shifting each letter 3 positions backward in the alphabet.

For Letter: H

Position of H in the alphabet: 8

Subtract the shift value: 8−3=58 - 3 = 58−3=5 The letter at position 5 is E.

Result so far: E

For Letter: Y

Position of Y in the alphabet: 25

Subtract the shift value: 25−3=2225 - 3 = 2225−3=22 The letter at position 22 is V.

Result so far: EV

For Letter: D

Position of D in the alphabet: 4

Subtract the shift value: 4−3=14 - 3 = 14−3=1

The letter at position 1 is A.

Result so far: EVA

For Letter: Q

Position of Q in the alphabet: 17

Subtract the shift value: 17−3=1417 - 3 = 1417−3=14 The letter at position 14 is N.

Result so far: EVAN

For Letter: L

Position of L in the alphabet: 12

Subtract the shift value: 12−3=912 - 3 = 912−3=9 The letter at position 9 is I.

Final Decoded Text: EVANI

Plaintext: EVANI
Encoded Ciphertext: HYDQL
Decoded Plaintext: EVANI

2.4 Key Considerations:

1. **Key**: The key in the Caesar cipher is the number by which the alphabet is shifted (e.g., 3 in the example). The key must remain secret, as knowing the key is necessary to decrypt the message.
2. **Security**: The Caesar cipher is very weak by modern standards because there are only 25 possible keys since shifting by 26 would return the original text). This makes it vulnerable to brute-force attacks.
3. **Breaking the Cipher**: An attacker can simply try all 25 shifts to break the cipher, which makes it easy to decipher without knowing the key (101computing, 2017).

2.5 Advantages

Table 1: Advantages

Simplicity	The cipher is simple to implement and doesn’t require complex algorithms or systems, making it an accessible introduction to cryptography.
Efficiency	The Caesar cipher requires minimal processing power, which allows for fast encryption and decryption, making it ideal for situations where resources are limited.
Historical Value	It has significant historical importance, being one of the earliest known encryption techniques, used by Julius Caesar for military communications.

2.6 Disadvantages

Table 2:Disadvantages

Limited Key space	The cipher is weak because it only allows for 25 possible keys, making it vulnerable to brute-force attacks, where an attacker can quickly try all possible shifts.
Easily Broken	By today’s standards, it offers minimal protection, easily defeated by modern cryptographic methods and analysis techniques like frequency analysis.
No Security Enhancements	The cipher doesn't include any modern encryption mechanisms, such as key exchange or multiple rounds of encryption, making it highly insecure by contemporary standards

3. Development of new Cryptographic Algorithm

Inverted Key Cipher (IKC):

The **Inverted Key Cipher (IKC)** is a method for encrypting and decrypting messages. It ensures that only someone with the correct key can read the original message. Here's how it works in a very simple and detailed way:

3.1 Working mechanism

In the Inverted Key Cipher (IKC), the key plays a critical role in the encryption process, and keeping it secret maintaining the security of the system. Here's how the mechanism works, assuming the key is kept secret:

1.Key Selection and Reversal: The process begins by selecting a key which can be a word or phrase that is kept confidential between the sender and receiver. The key is then reversed to add complexity. For instance, if the chosen key is "SECRET", it is reversed to "TERCES". This reversed key is used to shift the letters in the plaintext.

2.Shifting Process: Each letter of the plaintext is shifted according to the position of the corresponding letter in the reversed key. The position of the letter in the alphabet determines how many positions to shift the plaintext letter. For example, 'A' corresponds to 1, 'B' to 2, and so on. The corresponding letter of the reversed key provides the number of shifts.

If the shift exceeds 'Z', the alphabet wraps around using modulo 26 to ensure that the shift remains within the alphabet. The same shifting rule is applied for every character in the plaintext message, producing the ciphertext.

3.XOR Operation: After the shifting step, the ciphertext is subjected to an additional layer of security through the XOR operation. This involves converting both the ciphertext and the reversed key into their binary equivalents, and performing an XOR operation between the corresponding bits of each. This ensures that even if someone knows the ciphertext and the reversed key, without knowing the exact XOR step, it becomes more difficult to reverse-engineer the plaintext.

4.Security of the Cipher: This cipher remains secure as long as the key is kept secret. If the key is exposed, anyone who knows the key can easily reverse the process to decrypt the message. The strength of the IKC comes from the combination of the secret key, the reversal process, the shifting mechanism, and the XOR operation. Without access to the secret key, it is incredibly difficult for unauthorized parties to decipher the message.

3.2 Key Generation Algorithm for IKC

Step 1: Input the secret key phrase from the user.

Step 2: Validate the key phrase to ensure it contains only alphabetic characters and remove any non-alphabetic characters.

Step 3: Convert the key phrase to uppercase (optional for consistency).

Step 4: Reverse the key string to add complexity to the cipher.

Step 5: Assign each letter in the reversed key a numerical value based on its position in the alphabet (A=1, B=2, ..., Z=26).

Step 6: Store the reversed key string for future reference during decryption.

Step 7: Generate a list of shift values, each corresponding to the numerical value of the characters in the reversed key.

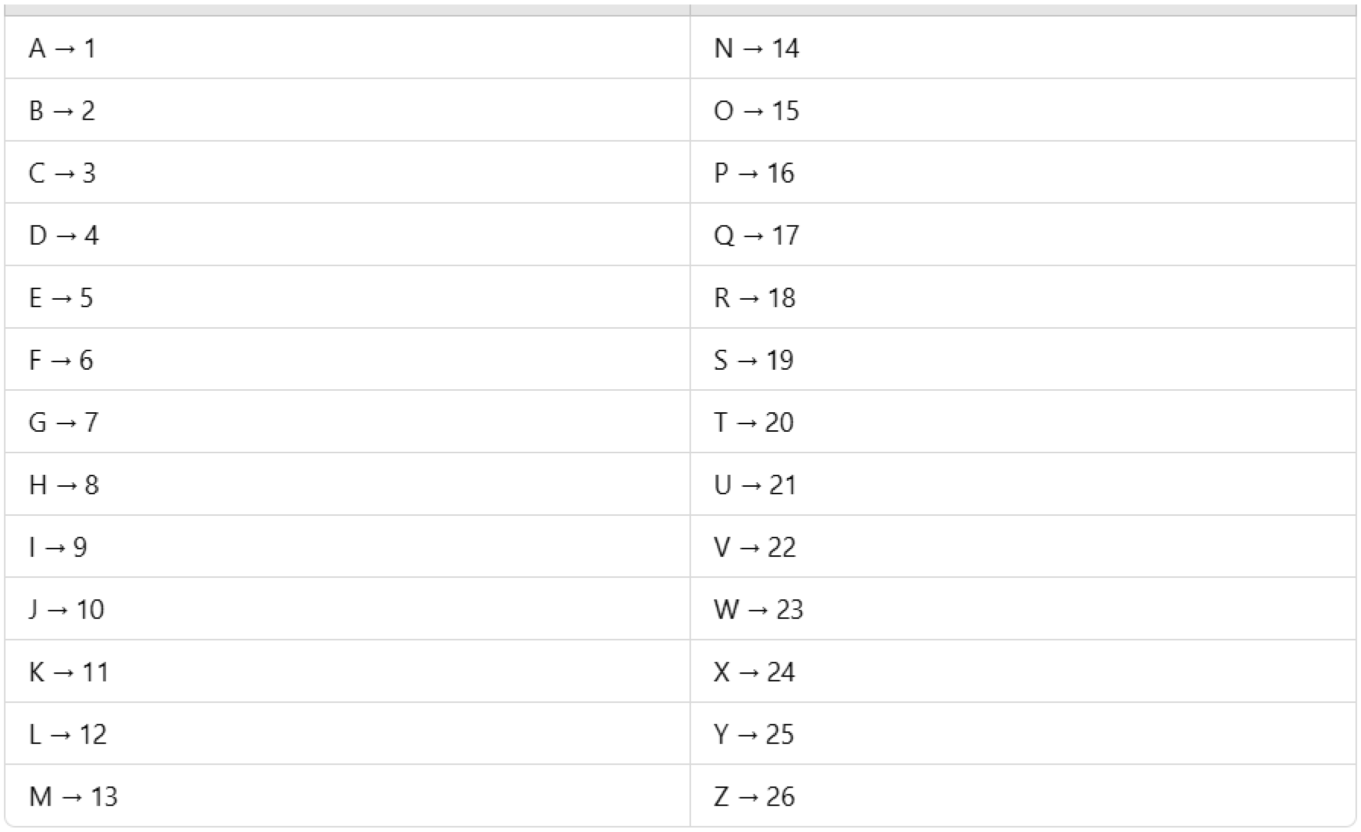
Step 8: Check the length of the key against the length of the plaintext. If the key is shorter, repeat the key until it matches the length of the plaintext.

Step 9: Securely store the key and its reverse, ensuring it is kept secret for encryption and decryption operations.

Example:

Key phrase = **ISLINGTON**
Reversed key = **NOTGNILSI**

Convert each letter in the reversed key to its corresponding numerical value (A=1, B=2, ..., Z=26):



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

Figure 13: Numeric Value

N = 14

O = 15

T = 20

G = 7

N = 14

I = 9

L = 12

S = 19

I = 9

Store the reversed key and corresponding shift values:

Reversed key: **NOTGNILSI**

Shift values: **14, 15, 20, 7, 14, 9, 12, 19, 9**

3.3 Encryption Algorithm for IKC

Step 1: Input the plaintext message from the user.

Step 2: Convert the plaintext to uppercase for consistency.

Step 3: Validate the plaintext to ensure it contains only alphabetic characters and remove any non-alphabetic characters.

Step 4: Convert each letter in the plaintext to its corresponding numerical value (A=1, B=2, ..., Z=26).

Step 5: Repeat the key (if necessary) to match the length of the plaintext.

Step 6: For each letter in the plaintext, apply the corresponding shift value from the key's list.

Step 7: Perform the shift operation. For example, add the key value to the plaintext letter’s numerical equivalent.

Step 8: If the shifted value exceeds 26, subtract 26 to ensure the result wraps around within the alphabet (e.g., 27 becomes 1).

Step 9: Convert the shifted numerical values back into letters.

Step 10: Combine the letters to form the ciphertext.

Step 11: Convert the ciphertext letters into their corresponding binary values.

Step 12: Convert the reversed key (used for shifting) into binary values as well, matching the length of the ciphertext.

Step 13: Perform the XOR operation between each binary value of the ciphertext and the corresponding binary value of the reversed key.

Step 14: The result of the XOR operation will give a new set of binary values.

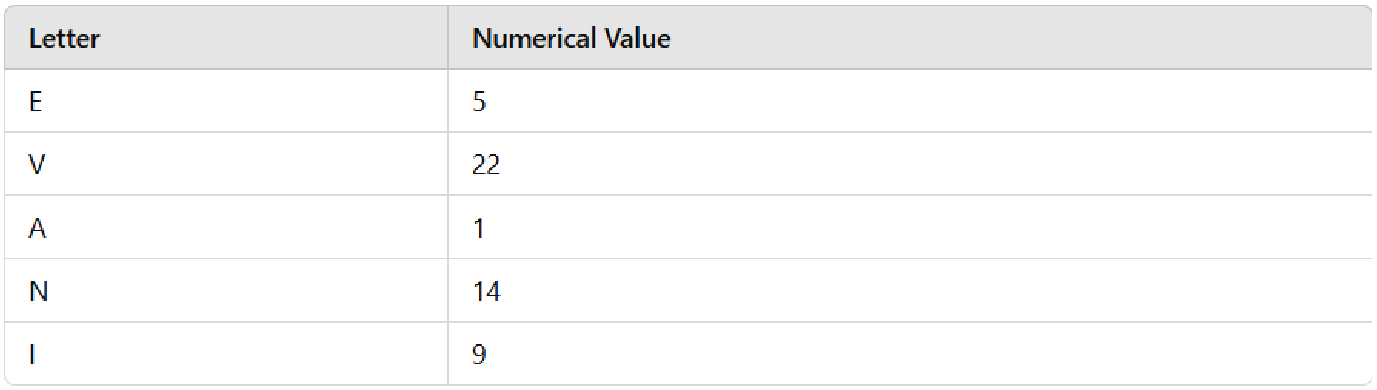
Step 15: Convert the XOR result back into the alphabetic characters.

Step 16: Combine the XORed letters to form the final encrypted ciphertext.

Example:

Plaintext = **EVANI**

Convert each letter in the plaintext to its corresponding numerical value (A=1, B=2, ..., Z=26):



Letter	Numerical Value
E	5
V	22
A	1
N	14
I	9

Figure 14: Numerical value

E = 5

V = 22

A = 1

N = 14

I = 9

Repeat the key (if necessary) to match the length of the plaintext:

Key (repeated): **NOTGNIL**

Shift values for encryption: **14, 15, 20, 7, 14**

Apply the shift for each letter in the plaintext:

For 'E' (5): Shift by 14 → 5 + 14 = 19 → 'S'

For 'V' (22): Shift by 15 → 22 + 15 = 37 → 37 - 26 = 11 → 'K'

For 'A' (1): Shift by 20 → 1 + 20 = 21 → 'U'

For 'N' (14): Shift by 7 → 14 + 7 = 21 → 'U'

For 'I' (9): Shift by 14 → 9 + 14 = 23 → 'W'

Combine the shifted letters:

Ciphertext = **SKUUW**

Steps to Convert Ciphertext into Binary Using XOR (Updated with XOR Step):

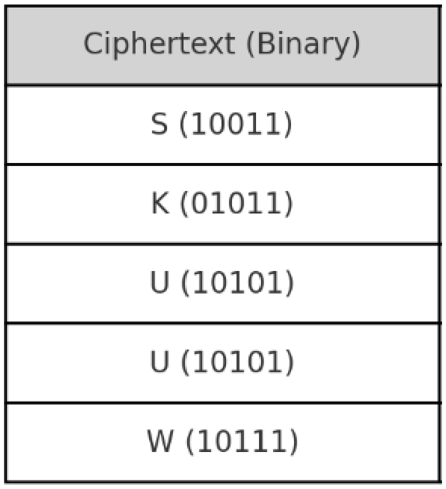
1.Ciphertext to Numerical Values:

Convert each letter in the ciphertext "SKUUW" to its corresponding numerical value (A=1, B=2, ..., Z=26):

S = 19, K = 11, U = 21, U = 21, W = 23

2.Convert Numerical Values to Binary:

Represent each numerical value using 5-bit binary:

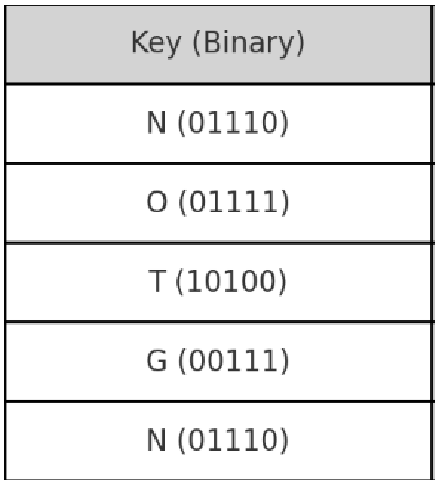


Ciphertext (Binary)
S (10011)
K (01011)
U (10101)
U (10101)
W (10111)

Figure 15:Numerical to Binary

S = 19 → 10011, **K** = 11 → 01011, **U** = 21 → 10101, **U** = 21 → 10101, **W** = 23 → 10111

1. **Key Conversion**:
   Reverse the key "ISLINGTON" to "NOTGNIL".
   Convert the reversed key into binary (using only the first five characters to match the ciphertext length):

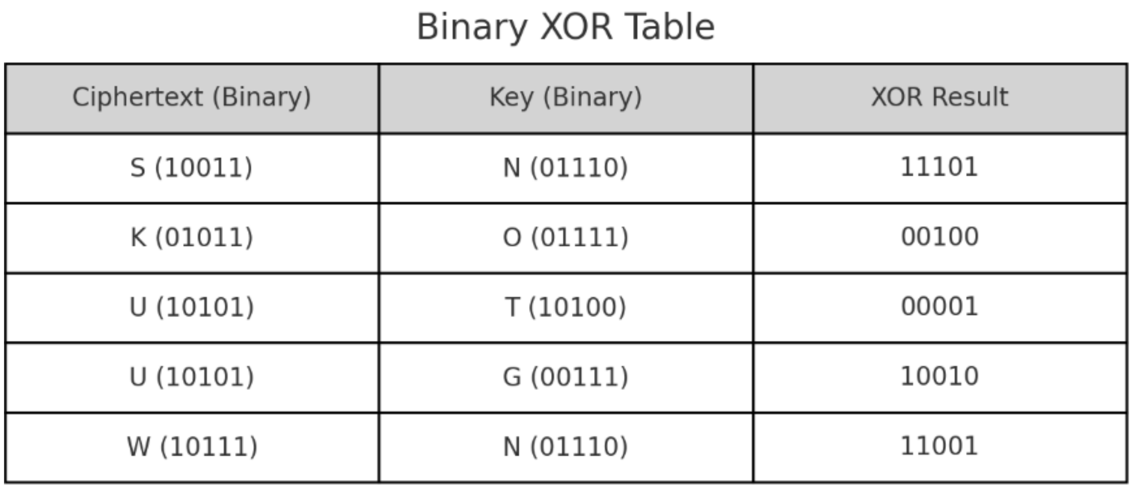


Key (Binary)
N (01110)
O (01111)
T (10100)
G (00111)
N (01110)

Figure 16: Key conversion

N = 14 → 01110, **O** = 15 → 01111, **T** = 20 → 10100, **G** = 7 → 00111, **N** = 14 → 01110

2. **XOR Operation** (New Step):
   Perform XOR between each binary representation of the ciphertext and the corresponding binary value of the key:



Binary XOR Table

Ciphertext (Binary)	Key (Binary)	XOR Result
S (10011)	N (01110)	11101
K (01011)	O (01111)	00100
U (10101)	T (10100)	00001
U (10101)	G (00111)	10010
W (10111)	N (01110)	11001

Figure 17: Binary XOR Table

- S (10011) XOR N (01110) = 11101
- K (01011) XOR O (01111) = 00100
- U (10101) XOR T (10100) = 00001
- U (10101) XOR G (00111) = 10010
- W (10111) XOR N (01110) = 11001

3. **Final Binary Representation**:

   Combine the XOR results to form the binary ciphertext:

   11101 00100 00001 10010 11001

3.4. Decryption Algorithm for IKC

Step 1: Input the ciphertext from the user.

Step 2: Convert the ciphertext to uppercase for consistency.

Step 3: Validate the ciphertext to ensure it contains only alphabetic characters and remove any non-alphabetic characters.

Step 4: Convert each letter in the ciphertext to its corresponding numerical value (A=1, B=2, ..., Z=26).

Step 5: Repeat the key (if necessary) to match the length of the ciphertext.

Step 6: Convert the reversed key (used for shifting) into binary (using the same length as the ciphertext).

Step 7: Perform the XOR operation between the binary values of the ciphertext and the corresponding binary values of the reversed key.

Step 8: Convert the XOR result from binary back into decimal values.

Step 9: Convert the decimal values back into their corresponding alphabetic characters.

Step 10: For each letter in the ciphertext, apply the reverse shift operation using the corresponding shift value from the key.

Step 11: Subtract the key value from the ciphertext letter's numerical equivalent to reverse the encryption process.

Step 12: If the result is less than 1, add 26 to ensure the result wraps around within the alphabet (e.g., 0 becomes 26).

Step 13: Convert the shifted numerical values back into letters.

Step 14: Combine the letters to form the decrypted plaintext.

Example:

Binary to Decimal: Convert the binary sequence back to decimal values:

$11101 \rightarrow 29$

00100 → 4

00001 → 1

10010 → 18

11001 → 25

Reverse the Key Conversion: The key is reversed as NOTGNIL. The corresponding decimal values are:

N = 14

O = 15

T = 20

G = 7

N = 14

XOR Decryption: Perform the XOR operation on the decrypted values:

29 (from the binary) XOR 14 (from the key) = 19 (S)

4 XOR 15 = 11 (K)

1 XOR 20 = 21 (U)

18 XOR 7 = 21 (U)

25 XOR 14 = 23 (W)

Ciphertext = **SKUUW**

Convert each letter in the ciphertext to its corresponding numerical value (A=1, B=2, ..., Z=26):

S = 19

K = 11

U = 21

U = 21

W = 23

Repeat the key (if necessary) to match the length of the ciphertext:

Key (repeated): **NOTGNIL**

Shift values for decryption: **14, 15, 20, 7, 14**

Apply the reverse shift for each letter in the ciphertext:

For 'S' (19): Reverse shift by 14 → 19 - 14 = 5 → 'E'

For 'K' (11): Reverse shift by 15 → 11 - 15 = -4 → -4 + 26 = 22 → 'V'

For 'U' (21): Reverse shift by 20 → 21 - 20 = 1 → 'A'

For 'U' (21): Reverse shift by 7 → 21 - 7 = 14 → 'N'

For 'W' (23): Reverse shift by 14 → 23 - 14 = 9 → 'I'

Combine the letters to form the decrypted plaintext:

Decrypted plaintext = **EVANI**

Summary

Key: ISLINGTON

Plaintext: EVANI

Ciphertext: SKUUW

Decrypted Plaintext: EVANI

3.5. Flowchart for Key Generation:

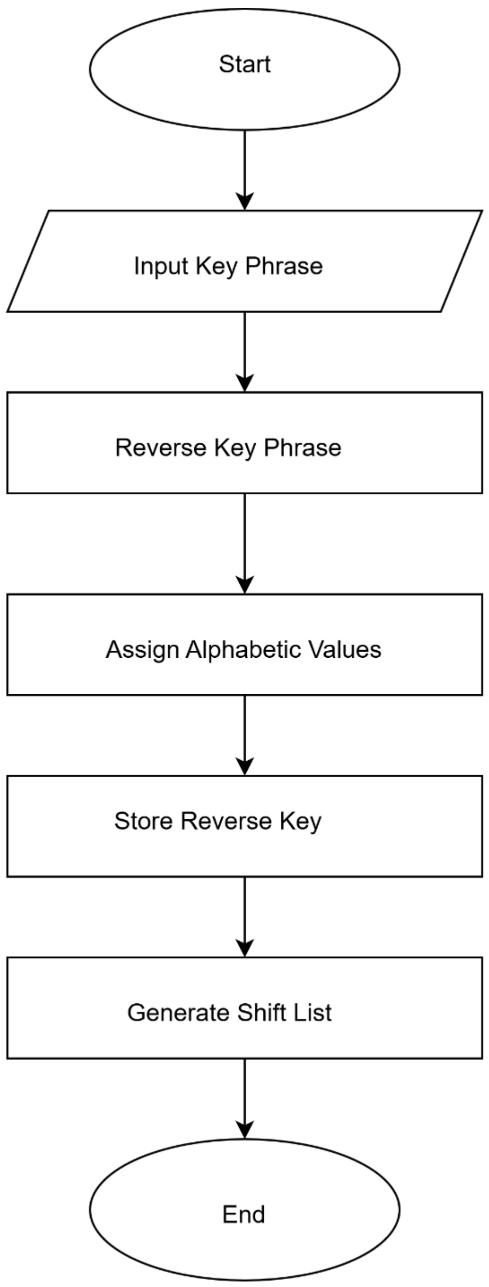


Figure 18: Flowchart 1

3.6. Flowchart for Encryption

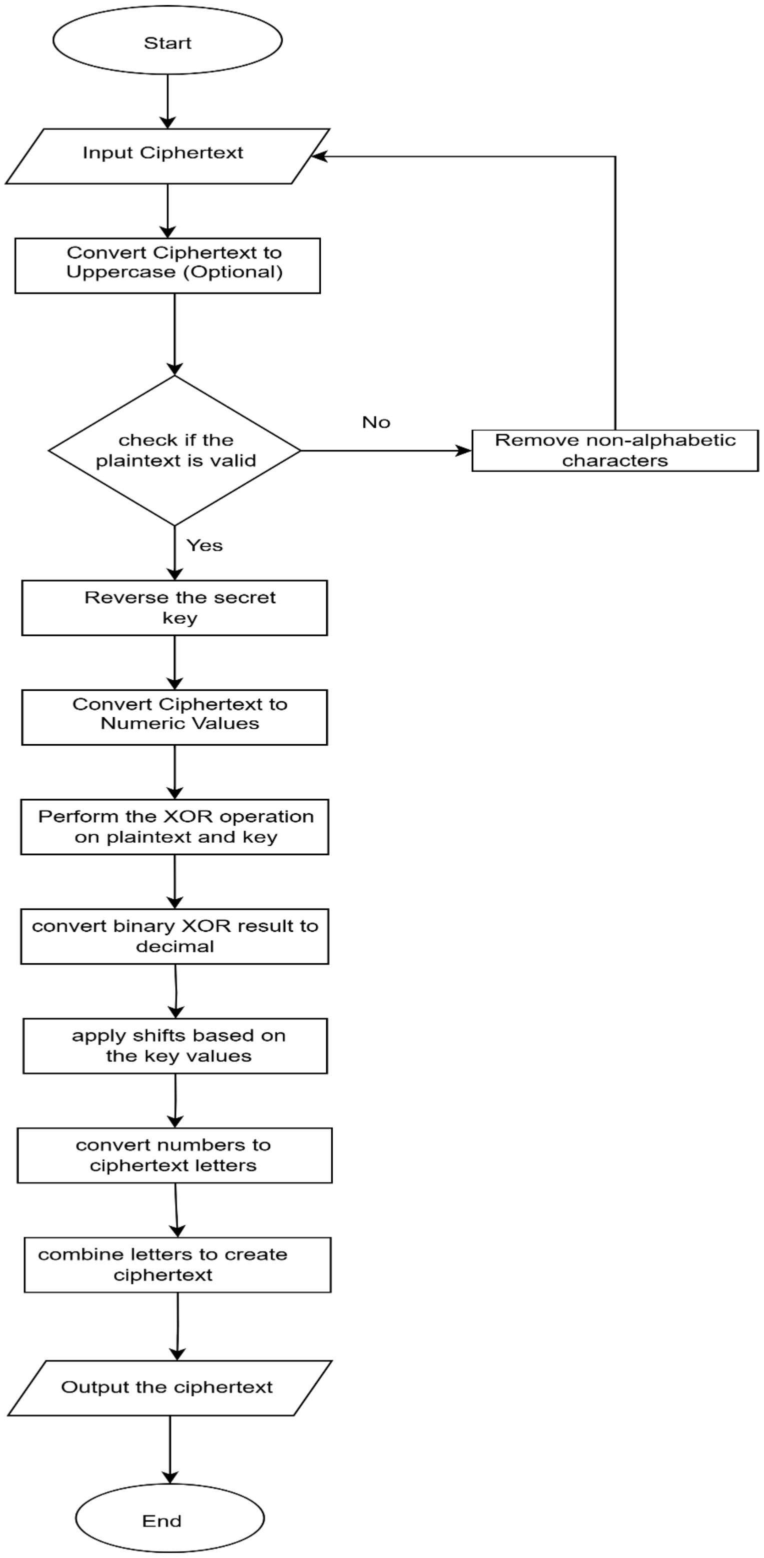

Start
Input Ciphertext
Convert Ciphertext to Uppercase (Optional)
check if the plaintext is valid
No
Remove non-alphabetic characters
Yes
Reverse the secret key
Convert Ciphertext to Numeric Values
Perform the XOR operation on plaintext and key
convert binary XOR result to decimal
apply shifts based on the key values
convert numbers to ciphertext letters
combine letters to create ciphertext
Output the ciphertext
End

Figure 19: Flowchart 2

3.7. Flowchart for Decryption

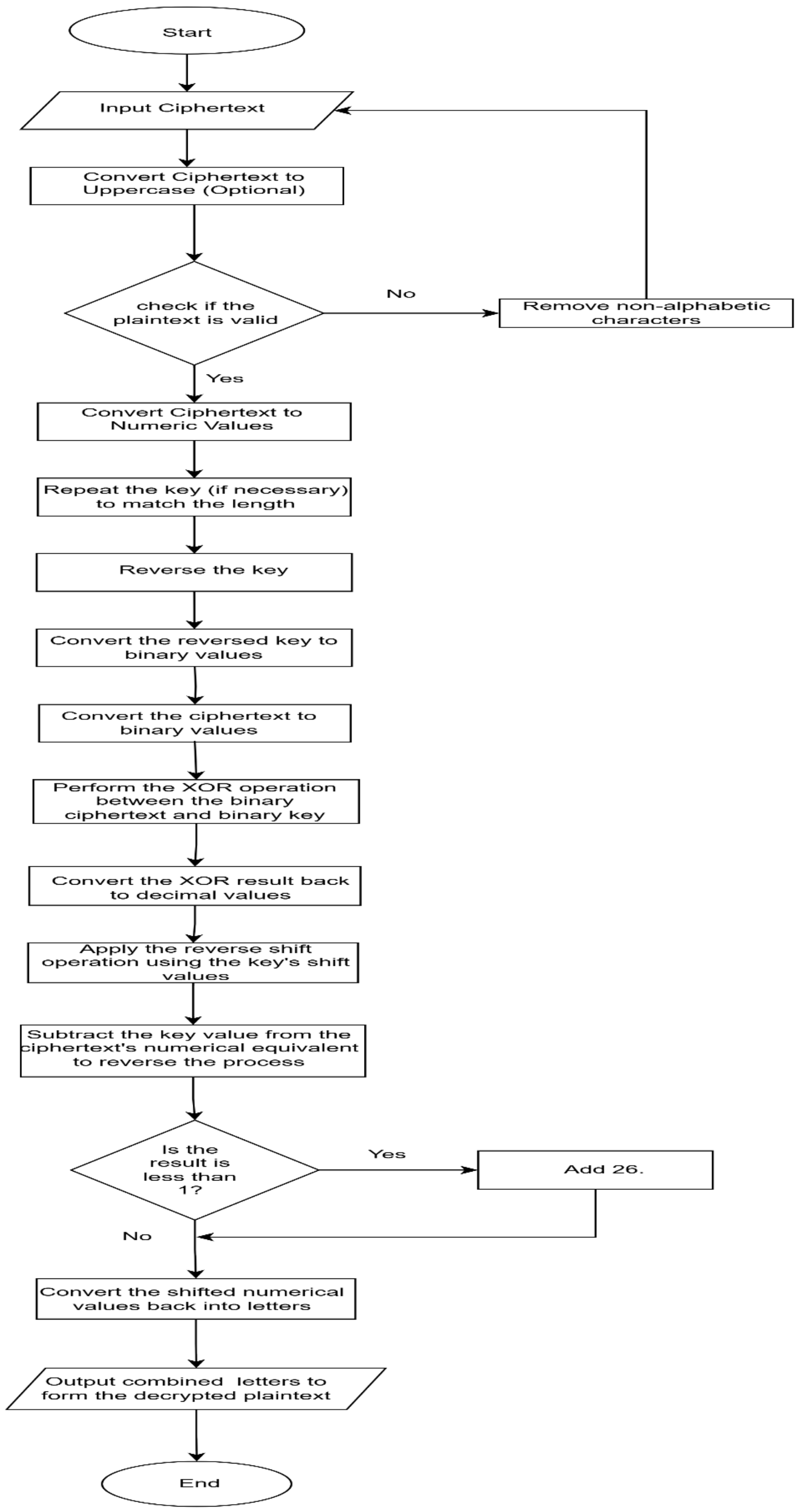

Start
Input Ciphertext
Convert Ciphertext to Uppercase (Optional)
check if the plaintext is valid
No
Remove non-alphabetic characters
Yes
Convert Ciphertext to Numeric Values
Repeat the key (if necessary) to match the length
Reverse the key
Convert the reversed key to binary values
Convert the ciphertext to binary values
Perform the XOR operation between the binary ciphertext and binary key
Convert the XOR result back to decimal values
Apply the reverse shift operation using the key's shift values
Subtract the key value from the ciphertext's numerical equivalent to reverse the process
Is the result is less than 1?
Yes
Add 26.
No
Convert the shifted numerical values back into letters
Output combined letters to form the decrypted plaintext
End

Figure 20: Flowchart 3

3.8. Necessity of modification

The **modification was necessary** because traditional cryptographic methods, such as classic ciphers, may not meet the security demands of modern applications. As computational power increases, traditional encryption techniques, like the Caesar cipher, become vulnerable to brute-force attacks and advanced decryption methods. These older ciphers are relatively simple and can be easily cracked with modern technology. Additionally, many classic encryption methods rely on static or predictable keys, which can be easily compromised. To address these concerns, the new methodology incorporates dynamic and complex key-generation techniques, which provide more secure encryption. Furthermore, as cyber threats continue to evolve, there is a need for encryption systems that are more adaptable to different environments and data types. The new approach ensures stronger and more unpredictable encryption patterns, making it more resilient to modern threats and harder to crack.

3.9. New methodology

The new methodology focuses on flexibility, security, and adaptability in cryptographic systems. It introduces several key improvements to traditional encryption techniques to enhance overall security. One of the most important aspects of the new methodology is dynamic key generation. Rather than relying on static or easily predictable keys, this method reverses a chosen key phrase and applies it to create a shifting pattern, making the encryption much more difficult to break. Another critical innovation is position-based shifting. This shifting mechanism changes depending on the position of each character in the message, as opposed to simply using a fixed shift. This adds an additional layer of complexity, making the encryption more resistant to attacks. The methodology also incorporates multiple layers of transformation, such as key reversal and position-based shifts, which ensure that the encryption process is far more intricate and secure compared to traditional methods.

3.10. New encryption algorithm

The new encryption algorithm combines traditional cryptographic methods with more advanced techniques to improve security and complexity. The process begins with the reversal of the user’s secret key (for example, "islington") to add an extra level of complexity. After reversing the key, each letter of the reversed key is assigned a number based on its position in the alphabet (A=1, B=2, C=3, etc.). These numeric values are then used to generate a list of shifts, which dictates how each letter in the plaintext will be transformed during encryption. Instead of using a single, fixed shift for all letters, each letter’s shift is determined by its corresponding position in the key’s shift list. This makes the cipher much harder to crack, as each letter is treated differently. The algorithm also ensures that if a shift exceeds 'Z', it wraps around to 'A', maintaining the integrity of the alphabet and ensuring the encryption remains consistent. Additionally, an XOR operation is applied to the binary representation of the plaintext and the key, adding complexity and making the ciphertext more unpredictable. The algorithm also ensures that shifts wrap around from 'Z' to 'A', maintaining consistency. This approach results in a secure ciphertext that cannot be easily decrypted without the original key.

3.11. The decryption algorithm

It reverses the encryption process, allowing the original plaintext to be recovered. The process begins by inputting the ciphertext that was generated during encryption. Then, for each letter in the ciphertext, the reverse shift is applied using the corresponding position in the key. Instead of shifting forward as in encryption, the key values are subtracted from the ciphertext letters' numerical equivalents. If the result of the subtraction is less than 1, the value is adjusted by wrapping around to the end of the alphabet (e.g., 0 becomes 26). After applying the reverse shifts to each letter, the numeric values are converted back into their corresponding letters, thereby reconstructing the original plaintext. The key used for decryption is the same as the one applied during encryption, ensuring the integrity and accuracy of the recovery process. The symmetry between the encryption and decryption steps ensures that without the key, it is nearly impossible to reverse the transformations and recover the original message.

4. Testing:

Here, I ran five test cases for the Inverted Key Cipher (IKC) algorithm, using the names of 5 ancient civilizations which are Atlantis, Babylon, Delphi, Thebes, and Persepolis as the plaintexts, and paired each one with a zodiac sign Scorpio, Leo, Sagittarius, Capricorn, and Taurus as the keys which helped me demonstrate the encryption and decryption process for each of the cases.

4.1Test 1:

Plaintext: “Atlantis” and Key: “Scorpio”

For Encryption Process

1. Reverse the Original Key

Original Key: Scorpio

Reversed Key: OiprocS

2. Assign Numerical Values to the Reversed Key

O = 15

I = 9

P = 16

R = 18

O = 15

C = 3

S = 19

3 Assign Numerical Values to the Plaintext

A = 1

T = 20

L = 12

A = 1

N = 14

T = 20

I = 9

S = 19

4. Match the Key to Plaintext Length

Reversed Key Repeated: OIPROCSO

Shift Values: 15, 9, 16, 18, 15, 3, 19, 15

5 Apply Shifts to Generate Ciphertext

A (1) + 15 = 16 → P

T (20) + 9 = 29 → (29 mod 26 = 3) → C

L (12) + 16 = 28 → (28 mod 26 = 2) → B

A (1) + 18 = 19 → S

N (14) + 15 = 29 → (29 mod 26 = 3) → C

T (20) + 3 = 23 → W

I (9) + 19 = 28 → (28 mod 26 = 2) → B

S (19) + 15 = 34 → (34 mod 26 = 8) → H

Ciphertext: PCBSCWBH

6 Binary Conversion and XOR

Ciphertext to Binary

P = 10000

C = 00011

B = 00010

S = 10011

C = 00011

W = 10110

B = 00010

H = 01000

Key to Binary

O = 01111

I = 01001

P = 10000

R = 10010

O = 01111

C = 00011

S = 10011

O = 01111

XOR Operation

P (10000) XOR O (01111) = 11111

C (00011) XOR I (01001) = 01010

B (00010) XOR P (10000) = 10010

S (10011) XOR R (10010) = 00001

C (00011) XOR O (01111) = 01100

W (10110) XOR C (00011) = 10101

B (00010) XOR S (10011) = 10001

H (01000) XOR O (01111) = 00111

Final XOR Output: 11111 01010 10010 00001 01100 10101 10001 00111

For Decryption Process

1 Reverse XOR Operation

Use the binary XOR results to retrieve the original ciphertext binary

11111 XOR 01111 = 10000 → P

01010 XOR 01001 = 00011 → C

10010 XOR 10000 = 00010 → B

00001 XOR 10010 = 10011 → S

01100 XOR 01111 = 00011 → C

10101 XOR 00011 = 10110 → W

10001 XOR 10011 = 00010 → B

00111 XOR 01111 = 01000 → H

2 Retrieve Numerical Values of Ciphertext

P = 16

C = 3

B = 2

S = 19

C = 3

W = 23

B = 2

H = 8

3 Key Matching and Shift Reversal

Reversed Key: OIPROCSO

Shift Values: 15, 9, 16, 18, 15, 3, 19, 15

Subtract the shift values from the ciphertext

P (16) - 15 = 1 → A

C (3) - 9 = -6 → (26 - 6 = 20) → T

B (2) - 16 = -14 → (26 - 14 = 12) → L

S (19) - 18 = 1 → A

C (3) - 15 = -12 → (26 - 12 = 14) → N

W (23) - 3 = 20 → T

B (2) - 19 = -17 → (26 - 17 = 9) → I

H (8) - 15 = -7 → (26 - 7 = 19) → S

4 Reconstructed Plaintext

ATLANTIS

4.2Test 2:

Plaintext "Babylon" and Key "Leo" For

Encryption Process

1. Reverse the Original Key

Original Key: Leo

Reversed Key: oeL

2. Assign Numerical Values to the Reversed Key

o = 15

e = 5

L = 12

3. Assign Numerical Values to the Plaintext

B = 2

A = 1

B = 2

Y = 25

L = 12

O = 15

N = 14

4. Match the Key to Plaintext Length

Reversed Key Repeated: oeLoeLo

Shift Values: 15, 5, 12, 15, 5, 12, 15

5. Apply Shifts to Generate Ciphertext

B (2) + 15 = 17 → Q

A (1) + 5 = 6 → F

B (2) + 12 = 14 → N

Y (25) + 15 = 40 → (40 mod 26 = 14) → N

L (12) + 5 = 17 → Q

O (15) + 12 = 27 → (27 mod 26 = 1) → A

N (14) + 15 = 29 → (29 mod 26 = 3) → C

Ciphertext: QFNNQAC

6. Binary Conversion and XOR

Ciphertext to Binary

Q = 10000

F = 00110

N = 01110

N = 01110

Q = 10000

A = 00001

C = 00011

Key to Binary

o = 01111

e = 00101

L = 01100

o = 01111

e = 00101

L = 01100

o = 01111

XOR Operation

Q (10000) XOR o (01111) = 11111

F (00110) XOR e (00101) = 00011

N (01110) XOR L (01100) = 00010

N (01110) XOR o (01111) = 00001

Q (10000) XOR e (00101) = 10101

A (00001) XOR L (01100) = 01101

C (00011) XOR o (01111) = 01100

Final XOR Output: 11111 00011 00010 00001 10101 01101 01100
For Decryption Process

1. Reverse XOR Operation

Use the binary XOR results to retrieve the original ciphertext binary

11111 XOR 01111 = 10000 → Q

00011 XOR 00101 = 00110 → F

00010 XOR 01100 = 01110 → N

00001 XOR 01111 = 01110 → N

10101 XOR 00101 = 10000 → Q

01101 XOR 01100 = 00001 → A

01100 XOR 01111 = 00011 → C

2. Retrieve Numerical Values of Ciphertext

Q = 16

F = 6

N = 14

N = 14

Q = 16

A = 1

C = 3

3. Key Matching and Shift Reversal

Reversed Key: oeLoeLo

Shift Values: 15, 5, 12, 15, 5, 12, 15

4. Subtract the shift values from the ciphertext

Q (16) - 15 = 1 → B

F (6) - 5 = 1 → A

N (14) - 12 = 2 → B

N (14) - 15 = -1 → (26 - 1 = 25) → Y

Q (16) - 5 = 11 → L

A (1) - 12 = -11 → (26 - 11 = 15) → O

C (3) - 15 = -12 → (26 - 12 = 14) → N

5. Reconstructed Plaintext

BABYLON

4.3Test 3:

Plaintext "Delphi" and Key "Sagittarius"

For Encryption Process

1.Reverse the Original Key

Original Key: Sagittarius

Reversed Key: suirattigaS

2. Assign Numerical Values to the Reversed Key

s = 19

u = 21

i = 9

r = 18

a = 1

t = 20

t = 20

i = 9

g = 7

a = 1

S = 19

3. Assign Numerical Values to the Plaintext

D = 4

E = 5

L = 12

P = 16

H = 8

I = 9

4. Match the Key to Plaintext Length

Reversed Key Repeated: suirattigaS

Shift Values: 19, 21, 9, 18, 1, 20

5. Apply Shifts to Generate Ciphertext

D (4) + 19 = 23 → W

E (5) + 21 = 26 → (26 mod 26 = 0) → A

L (12) + 9 = 21 → U

P (16) + 18 = 34 → (34 mod 26 = 8) → H

H (8) + 1 = 9 → I

I (9) + 20 = 29 → (29 mod 26 = 3) → C

Ciphertext: WAUHIC

6. Binary Conversion and XOR

Ciphertext to Binary

W = 10111

A = 00001

U = 10101

H = 01000

I = 01001

C = 00011

Key to Binary

s = 01111

u = 01011

i = 01001

r = 10010

a = 00001

t = 10100

XOR Operation

W (10111) XOR s (01111) = 11000

A (00001) XOR u (01011) = 01010

U (10101) XOR i (01001) = 11100

H (01000) XOR r (10010) = 11010

I (01001) XOR a (00001) = 01000

C (00011) XOR t (10100) = 10111

Final XOR Output: 11000 01010 11100 11010 01000 10111

Decryption Process

1. Reverse XOR Operation

Use the binary XOR results to retrieve the original ciphertext binary

11000 XOR 01111 = 10111 → W

01010 XOR 01011 = 00001 → A

11100 XOR 01001 = 10101 → U

11010 XOR 10010 = 01000 → H

01000 XOR 00001 = 01001 → I

10111 XOR 10100 = 00011 → C

2. Retrieve Numerical Values of Ciphertext

W = 23

A = 1

U = 21

H = 8

I = 9

C = 3

3. Key Matching and Shift Reversal

Reversed Key: suirattigaS

Shift Values: 19, 21, 9, 18, 1, 20

4. Subtract the shift values from the ciphertext

W (23) - 19 = 4 → D

A (1) - 21 = -20 → (26 - 20 = 6) → E

U (21) - 9 = 12 → L

H (8) - 18 = -10 → (26 - 10 = 16) → P

I (9) - 1 = 8 → H

C (3) - 20 = -17 → (26 - 17 = 9) → I

5. Reconstructed Plaintext

DELPHI

4.4 Test 4:

Plaintext "Thebes" and Key "Capricorn"

For Encryption Process

1. Reverse the Original Key

Original Key: Capricorn

Reversed Key: niorpacaC

2. Assign Numerical Values to the Reversed Key

n = 14

i = 9

o = 15

r = 18

p = 16

a = 1

c = 3

a = 1

C = 3

3. Assign Numerical Values to the Plaintext

T = 20

H = 8

E = 5

B = 2

E = 5

S = 19

4. Match the Key to Plaintext Length

Reversed Key Repeated: niorpaca

Shift Values: 14, 9, 15, 18, 16, 1

5. Apply Shifts to Generate Ciphertext

T (20) + 14 = 34 → (34 mod 26 = 8) → H

H (8) + 9 = 17 → Q

E (5) + 15 = 20 → T

B (2) + 18 = 20 → T

E (5) + 16 = 21 → U

S (19) + 1 = 20 → T

Ciphertext: HQTUTT

6. Binary Conversion and XOR

Ciphertext to Binary

H = 01000

Q = 10000

T = 10100

T = 10100

U = 10101

T = 10100

Key to Binary

n = 01110

i = 01001

o = 01111

r = 10010

p = 10000

a = 00001

XOR Operation

H (01000) XOR n (01110) = 00110

Q (10000) XOR i (01001) = 11001

T (10100) XOR o (01111) = 11011

T (10100) XOR r (10010) = 00110

U (10101) XOR p (10000) = 00101

T (10100) XOR a (00001) = 10101

Final XOR Output: 00110 11001 11011 00110 00101 10101

For Decryption Process

1. **Reverse XOR Operation**

Use the binary XOR results to retrieve the original ciphertext binary

00110 XOR 01110 = 01000 → H

11001 XOR 01001 = 10000 → Q

11011 XOR 01111 = 10100 → T

00110 XOR 10010 = 10100 → T

00101 XOR 10000 = 10101 → U

10101 XOR 00001 = 10100 → T

2. Retrieve Numerical Values of Ciphertext

H = 8

Q = 17

T = 20

T = 20

U = 21

T = 20

3. Key Matching and Shift Reversal

Reversed Key: niorpaca

Shift Values: 14, 9, 15, 18, 16, 1

4. Subtract the shift values from the ciphertext

H (8) - 14 = -6 → (26 - 6 = 20) → T

Q (17) - 9 = 8 → H

T (20) - 15 = 5 → E

T (20) - 18 = 2 → B

U (21) - 16 = 5 → E

T (20) - 1 = 19 → S

5. Reconstructed Plaintext

THEBES

4.5 Test 5:

Plaintext "Persepolis" and Key "Taurus"

For Encryption Process

1. Reverse the Original Key

Original Key: Taurus

Reversed Key: suaruT

2. Assign Numerical Values to the Reversed Key

s = 19

u = 21

a = 1

r = 18

u = 21

T = 20

3. Assign Numerical Values to the Plaintext

P = 16

E = 5

R = 18

S = 19

E = 5

P = 16

O = 15

L = 12

I = 9

S = 19

4. Match the Key to Plaintext Length

Reversed Key Repeated: suaruTs

Shift Values: 19, 21, 1, 18, 21, 20, 19, 21, 1, 18

5. Apply Shifts to Generate Ciphertext

P (16) + 19 = 35 → (35 mod 26 = 9) → I

E (5) + 21 = 26 → (26 mod 26 = 0) → A

R (18) + 1 = 19 → S

S (19) + 18 = 37 → (37 mod 26 = 11) → K

E (5) + 21 = 26 → (26 mod 26 = 0) → A

P (16) + 20 = 36 → (36 mod 26 = 10) → J

O (15) + 19 = 34 → (34 mod 26 = 8) → H

L (12) + 21 = 33 → (33 mod 26 = 7) → G

I (9) + 1 = 10 → J

S (19) + 18 = 37 → (37 mod 26 = 11) → K

Ciphertext: IASKAJHGJ

6.Binary Conversion and XOR

Ciphertext to Binary

I = 01001

A = 00001

S = 10011

K = 10100

A = 00001

J = 01010

H = 01000

G = 00111

J = 01010

K = 10100

Key to Binary

s = 01111

u = 01011

a = 00001

r = 10010

u = 01011

T = 10100

XOR Operation

I (01001) XOR s (01111) = 00110

A (00001) XOR u (01011) = 01010

S (10011) XOR a (00001) = 10010

K (10100) XOR r (10010) = 00110

A (00001) XOR u (01011) = 01010

J (01010) XOR T (10100) = 11110

H (01000) XOR s (01111) = 00111

G (00111) XOR u (01011) = 01100

J (01010) XOR a (00001) = 01011

K (10100) XOR r (10010) = 00110

Final XOR Output: 00110 01010 10010 00110 01010 11110 00111 01100 01011 00110

For Decryption Process

1. **Reverse XOR Operation**

Use the binary XOR results to retrieve the original ciphertext binary

00110 XOR 01111 = 01001 → I

01010 XOR 01011 = 00001 → A

10010 XOR 00001 = 10011 → S

00110 XOR 10010 = 10100 → K

01010 XOR 01011 = 00001 → A

11110 XOR 10100 = 01010 → J

00111 XOR 01111 = 01000 → H

01100 XOR 01011 = 00111 → G

01011 XOR 00001 = 01010 → J

00110 XOR 10010 = 10100 → K

2. **Retrieve Numerical Values of Ciphertext**

I = 9

A = 1

S = 19

K = 11

A = 1

J = 10

H = 8

G = 7

J = 10

K = 11

3. **Key Matching and Shift Reversal**

Reversed Key: suaruTs

Shift Values: 19, 21, 1, 18, 21, 20, 19, 21, 1, 18

4. **Subtract the shift values from the ciphertext**

I (9) - 19 = -10 → (26 - 10 = 16) → P

A (1) - 21 = -20 → (26 - 20 = 6) → E

S (19) - 1 = 18 → R

K (11) - 18 = -7 → (26 - 7 = 19) → S

A (1) - 21 = -20 → (26 - 20 = 6) → P

J (10) - 20 = -10 → (26 - 10 = 16) → O

H (8) - 19 = -11 → (26 - 11 = 15) → L

G (7) - 21 = -14 → (26 - 14 = 12) → I

J (10) - 1 = 9 → S

K (11) - 18 = -7 → (26 - 7 = 19) → S

5. Reconstructed Plaintext

PERSEPOLIS

5.Critical Evaluation of the new Cryptographic Algorithm

This section provides a detailed evaluation of the Inverted Key Cipher (IKC) algorithm designed in TASK 03. It explores the strengths and weaknesses of the algorithm while also identifying areas where it can be practically applied. By examining its unique features, this analysis aims to assess how well the IKC algorithm meets the demands of real-world cryptographic challenges.

5.1.Strengths of the IKC Algorithm

Enhanced Complexity with Key Reversal:

By reversing the key, the algorithm introduces an added layer of complexity, making it more resistant to simple cryptanalysis methods.

Dynamic Shifting Mechanism:

The algorithm uses unique shift values for each letter, derived from the reversed key. This approach prevents patterns in the plaintext from directly mapping to predictable patterns in the ciphertext.

Integration of XOR for Added Security:

Incorporating a binary XOR operation with the reversed key adds a secondary encryption layer, significantly strengthening the algorithm’s defenses against brute force and statistical attacks.

Seamless Alphabet Wrapping:

The use of modular arithmetic ensures all transformations remain within the bounds of the alphabet, maintaining accuracy and consistency during both encryption and decryption.

Adaptability to Key Lengths:

The ability to repeat the key to match the length of the plaintext or ciphertext makes the algorithm versatile, enabling it to handle messages of varying sizes efficiently.

5.2 Weaknesses of the IKC Algorithm

Reliance on Key Confidentiality:

The algorithm's security is highly dependent on keeping the key secret. If the key is exposed, the encryption becomes highly vulnerable to attacks.

Higher Computational Demand:

With multiple steps like key reversal, shift calculations, and XOR operations, the algorithm requires more processing power compared to simpler encryption techniques.

Potential Vulnerability to Advanced Cryptanalysis:

Despite its complexity, the algorithm could be susceptible to sophisticated attacks such as chosen plaintext or differential cryptanalysis, particularly if partial key or ciphertext patterns are discovered.

Lack of Built-In Message Authentication:

The IKC algorithm does not include mechanisms to verify the integrity or authenticity of the message, leaving it open to tampering.

Complexity of Binary Conversion:

The dependency on binary conversions for XOR operations adds an extra layer of complexity, increasing the risk of implementation errors, especially in systems with limited resources.

5.3 Application Areas of the IKC Algorithm

Secure Communication Platforms:

The IKC algorithm is ideal for use in messaging applications, ensuring that user conversations remain private and protected from unauthorized access.

File Protection:

Its robust encryption capabilities make it well-suited for securing sensitive files, particularly when transferring data over untrusted or public networks.

Embedded and IoT Systems:

With its adaptability, IKC is a good fit for resource-limited environments such as IoT devices, where secure data transmission is essential.

Short-Term Data Encryption:

The algorithm is effective for protecting information that requires temporary confidentiality, such as session keys or temporary access tokens.

Educational and Research Applications:

Given its innovative design, IKC can be utilized as a learning tool in cryptography courses, offering students hands-on experience with concepts like key reversal and dynamic shift mechanisms.

6.Conclusion

In today’s rapidly evolving digital world, the need for strong cryptographic systems is more crucial than ever. This report introduced the Inverted Key Cipher (IKC), a new encryption method designed to enhance security. By incorporating unique features like key reversal, dynamic shifting, and XOR operations, IKC aims to provide stronger protection against common cryptographic attacks, such as brute-force and statistical methods.

However, the IKC algorithm is not without its drawbacks. Its security is heavily dependent on the secrecy of the key if the key is compromised, the entire encryption system becomes vulnerable. Additionally, the algorithm requires more computational resources than traditional encryption methods, and while it offers better protection, it could still be vulnerable to advanced cryptanalysis. Moreover, it does not include a built-in system for verifying the integrity of the message, leaving data open to potential tampering.

Despite these challenges, IKC is well-suited for applications like secure communication platforms, file encryption, and IoT systems, where secure data transmission is vital. Its flexibility and educational value also make it a useful tool for cryptography students and researchers. While there are areas that need improvement, such as key management and computational efficiency, IKC presents a promising option for enhancing data security in today’s digital landscape.

7.Bibliography

(2023, June 10). Retrieved from thalesgroup: https://www.thalesgroup.com/en/markets/digital-identity-and-security/magazine/brief-history-encryption

(2024, Nov 18). Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/cryptography-and-its-types/

(2016, Nov 19). Retrieved from inviul: https://www.inviul.com/cryptosystem-and-its-types/

(2017, Jan 21). Retrieved from 101computing: https://www.101computing.net/caesar-cipher/

(2022, June 27). Retrieved from bundesdruckerei: https://www.bundesdruckerei.de/en/innovation-hub/history-cryptography

(2024, 12 10). Retrieved from dencode: https://dencode.com/cipher/scytale

(2024, Jun 27). Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/

(2024, Sept 1). Retrieved from argoox: https://argoox.com/blog/what-is-a-cryptosystem-in-cryptography/

(2024, Dec 10). Retrieved from allcryptowhitepapers: https://www.allcryptowhitepapers.com/what-is-cryptosystem/

(2024, May 28). Retrieved from cpl.thalesgroup: https://cpl.thalesgroup.com/blog/modern-cryptography-journey

(2024, Dec 10). Retrieved from informationsecurity: https://informationsecurity.wustl.edu/items/confidentiality-integrity-and-availability-the-cia-triad/

(2024, Dec 10). Retrieved from dlab.epfl: https://dlab.epfl.ch/wikispeedia/wpcd/wp/c/Caesar_cipher.htm

Anthony Henderson. (2024, Nov 23). Retrieved from panmore: https://panmore.com/the-cia-triad-confidentiality-integrity-availability

Gustavus J. Simmons. (2024 , Oct 20). Retrieved from britannica: https://www.britannica.com/topic/cryptology

Josh Lake. (2023, Nov 1). Retrieved from comparitech: https://www.comparitech.com/blog/information-security/confidentiality-integrity-availability/

Margaret Rouse. (2022, Dec 20). Retrieved from techopedia: https://www.techopedia.com/definition/25830/cia-triad-of-information-security

Mariana. (2024, Jan 27). Retrieved from oak.atlantia.sca: https://oak.atlantia.sca.org/a-brief-history-of-western-cryptography-through-the-middle-ages/

Nickpelling. (2008, Jun 01). Retrieved from ciphermysteries: https://ciphermysteries.com/2008/06/01/codes-and-ciphers-through-the-middle-ages

Ofir Kuperman. (2022, Nov 2). Retrieved from davidson.weizmann.ac.il: https://davidson.weizmann.ac.il/en/online/askexpert/codebreakers-caesar-cipher

Ron Ross. (2017, Nov 28). Retrieved from nist: https://www.nist.gov/blogs/taking-measure/why-security-and-privacy-matter-digital-world

SentinelOne. (2024, May 14). Retrieved from sentinelone: https://www.sentinelone.com/cybersecurity-101/cybersecurity/cia-triad/