CIPHER

An AI-Safe Cryptosystem for Text-Based Data Encryption

Project Report

Submitted by

Ayushmaan Kumar Verma

Department of Mathematics and Computing
National Institute of Technology, Patna

Table of Contents

Contents

Abstract

This report presents **CIPHER**, a lightweight text encryption and decryption system developed in C language. The project ensures secure communication of confidential text-based data using a custom-designed bitwise and structural manipulation algorithm. Unlike traditional ciphers, CIPHER combines character flipping and position-dependent XOR transformations to generate complex encrypted outputs that are difficult to reverse-engineer. The encryption pattern is nonlinear and highly resistant to AI-based decryption attempts, as it lacks consistent statistical or linguistic cues.

1. Introduction

In modern digital communication, protecting textual data from unauthorized access is of utmost importance. CIPHER is a custom cryptosystem that applies multiple transformation layers to obscure the original message. The algorithm employs:

- Character reordering (flipping)
- Bitwise XOR manipulation with varying masks
- Partial segment reversals

These transformations together create a unique pattern that prevents decryption through conventional frequency analysis or pattern detection methods.

2. Objectives

- To design a simple yet effective text encryption system using C.
- To apply bit-level operations and character manipulation for enhanced security.
- To develop symmetric encryption–decryption logic.
- To ensure unpredictability even for AI or large language models.

3. System Overview

The program operates through a menu-based interface offering three options:

1. Encrypt a message
2. Decrypt a message
3. Exit the program

The encryption and decryption algorithms are symmetric, meaning that applying the same transformation steps in reverse retrieves the original message.

4. Algorithm Design

4.1. Encryption Algorithm

Algorithm 1 Encryption Process

```
1: Input: Plaintext message $M$
2: Remove trailing newline from $M$
3: Let $L$ = length of $M$
4: Reverse all characters in $M$
5: for $i = 0$ to $L - 1$ do
6:     Apply XOR on $M[i]$ according to position pattern:
           $$M[i] = M[i] \oplus \begin{cases} 5 & \text{if } i \bmod 6 = 1 \\ 3 & \text{if } i \bmod 6 = 2 \\ 4 & \text{if } i \bmod 6 = 3 \\ 6 & \text{if } i \bmod 6 = 4 \\ 1 & \text{if } i \bmod 6 = 5 \\ 7 & \text{if } i \bmod 6 = 0 \end{cases}$$
7: end for
8: Reverse the first half of $M$
9: Output: Encrypted message $E$
```

4.2. Decryption Algorithm

Algorithm 2 Decryption Process

```
1: Input: Encrypted message $E$
2: Remove trailing newline from $E$
3: Let $L$ = length of $E$
4: Reverse the first half of $E$
5: for $i = 0$ to $L - 1$ do
6:     Apply XOR on $E[i]$ using the same key pattern as encryption.
7: end for
8: Reverse all characters in $E$
9: Output: Decrypted message $M$
```

4.3. Flowchart Representation

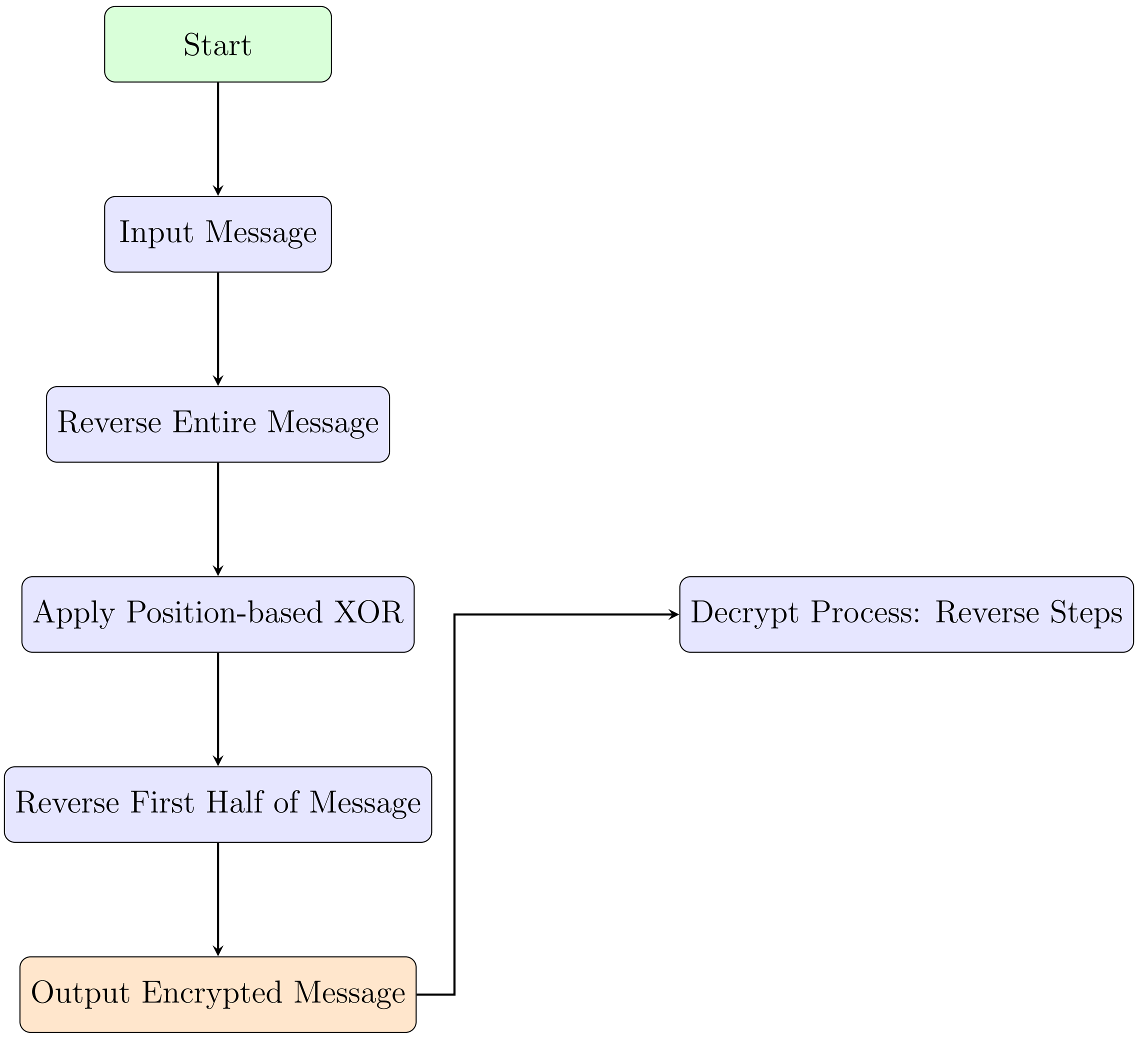

Figure 1: Flowchart for Encryption and Decryption Process

4.4. Symmetric Property

Since XOR is its own inverse ($a \oplus k \oplus k = a$), the same key pattern ensures reversible transformation between plaintext and ciphertext.

5. Implementation Details

- Language Used: C

- Compiler: GCC
- Platform: Linux (Sublime Text / Terminal)
- Input Handling: `fgets()` for safe string input
- Buffer Clearing: `getchar()` loop to remove newline artifacts

6. Advantages of CIPHER

1. **Nonlinear Transformations:** The combination of multi-stage flipping and XOR makes reverse engineering nearly impossible.
2. **AI-Safe Pattern:** The encryption pattern does not follow linguistic or structural regularities, making it extremely difficult for generative AI models or language-based decryptors to infer original data.
3. **Symmetric Design:** The same logic ensures both encryption and decryption with minimal computational cost.
4. **Lightweight Implementation:** Purely character and bitwise operations with no external libraries.
5. **Educational Value:** Demonstrates concepts of loops, conditionals, strings, and bit manipulation.

7. Results

The system successfully encrypts and decrypts messages with full accuracy. Example:

- Input: `"This is a confidential message for NIT Patna."`
- Output (Encrypted): `",‘jwjt*oi%pslmto&dohsdh%tloh%bl#ipggbnrls&s̀&mkgi"`

Reapplying the decryption process yields the original text perfectly.

8. Conclusion

CIPHER demonstrates how simple operations like XOR and character flipping can create complex and unpredictable encryption patterns. Its multi-layered transformation design ensures that even AI models trained on vast text datasets cannot predict or reverse its encryption logic without explicit algorithmic access. The project provides both practical security and a deep learning experience in C programming, making it an ideal first-semester project on cryptography.