

Sending Super Secret Messages with Book Cipher

Adrian Nogales, Juan Manuel Franco and Daniel Gomollón

1. Abstract

In an age of advanced encryption and digital security, this project revisits a historical cipher technique known as the Book Cipher. We implemented a Python-based solution that encrypts and decrypts messages using a shared book as the key. Each character in the plaintext message is replaced by its position in a book, and decryption uses these positions to recover the original text. This report explores the methodology, theory, and implementation used, making it an educational journey into classical cryptography with a modern twist.

2. Introduction

a. Aim

The goal of this project is to create a system that securely sends messages using the Book Cipher method-where both the sender and receiver share a copy of the same book. It encrypts a message by converting characters into position coordinates within the book and decrypts it back using those positions.

b. Scope

This project covers:

- Reading and indexing the book text.
- Encrypting messages using character positions from the book.
- Decrypting messages using the stored positions.
- Error handling for unknown characters and malformed inputs.
- Basic use of randomness to enhance security.

c. Methodology

The implementation is done using Python, specifically utilizing:

- Text parsing and string manipulation.
- Random module for selecting varied character positions.
- File I/O for reading the book text.
- Functions for modularity and clarity: `create_index`, `book_cipher`, and `book_decipher`.

3. Cryptographic Theory: Book Cipher

A Book Cipher is a type of code in which the sender and receiver use the same book to encrypt and decrypt messages. Each character in the message is represented by its position in the book-typically encoded as a combination of page, line, word, and letter number.

Advantages:

- Extremely difficult to break without the correct book.
- Simple to implement and understand.
- The book serves as a large and complex key.

Challenges:

- Requires both parties to have an identical copy of the book.
- Any textual inconsistency (e.g., different punctuation) can break decryption.
- Requires a method to handle characters not found in the book.

4. Implementation and Results

4.1 Indexing

The `create_index(book_text)` function parses the book into a list of character positions, skipping empty lines and question marks. The format is:

(page_number, row_number, word_number, letter_index, letter)

4.2 Encryption

The `book_cipher(plaintext, index)` function:

- Looks up all positions for each character.
- Randomly selects one of these positions.
- Converts the character to a position string (e.g., "1:5:3:2").
- Preserves spaces and question marks.
- Replaces characters not found in the book with '?'.

4.3 Decryption

The `book_decipher(ciphered_list, book_text)` function:

- Splits the book again.
- Extracts the original character from each given position.
- Handles errors and unknown characters by appending '?'.

4.4 Results

Both encryption and decryption worked effectively, even with random positions for each letter. The system is robust against most small errors and provides a good layer of obfuscation for casual communication.

5. Summary

The main goal of implementing a working Book Cipher system was successfully achieved. The script can encrypt and decrypt messages reliably, using a shared book as the key. It preserves formatting where possible and handles exceptions gracefully. One limitation is that decryption assumes a perfect match between the book versions used by sender and receiver.

Future improvements could include:

- Support for case sensitivity and punctuation.
- A deterministic mode for debugging.
- GUI or web interface for user interaction.

6. Bibliography

1. Wikipedia contributors. (2023). Book cipher. Wikipedia. https://en.wikipedia.org/wiki/Book_cipher
2. Wikipedia contributors. (2023). Plagiarism. Wikipedia. <https://en.wikipedia.org/wiki/Plagiarism>
3. Python Software Foundation. (2023). random - Generate pseudo-random numbers.

<https://docs.python.org/3/library/random.html>