Encryption Algorithm Competition

MD. Nabil Safowan [1] Mubasshira Siddiqua [2] Tanin Khan Porna[3] Shahnewaz Hossain Eshad [4]

Department of Computer Science and Engineering
Independent University, Bangladesh
Dhaka, Bangladesh.
{[1]2110488@iub.edu.bd,[2]2131278@iub.edu.bd,[3]2030267@iub.edu.bd,[4]2020069@iub.edu.bd

Abstract

The algorithm I have chosen and implemented for this project is a basic symmetric encryption technique called "XOR Cipher" or "XOR Encryption". XOR stands for "exclusive or". It is a logical operation and it takes two binary inputs. The output is true or 1 only when the bits are different.
Initially, I went for RSA algorithm for this project but due to restrictions being put in place regarding the length of the cipher string, I had to change our algorithm midway. Overall I achieved a decent score using the XOR encryption algorithm.

Introduction

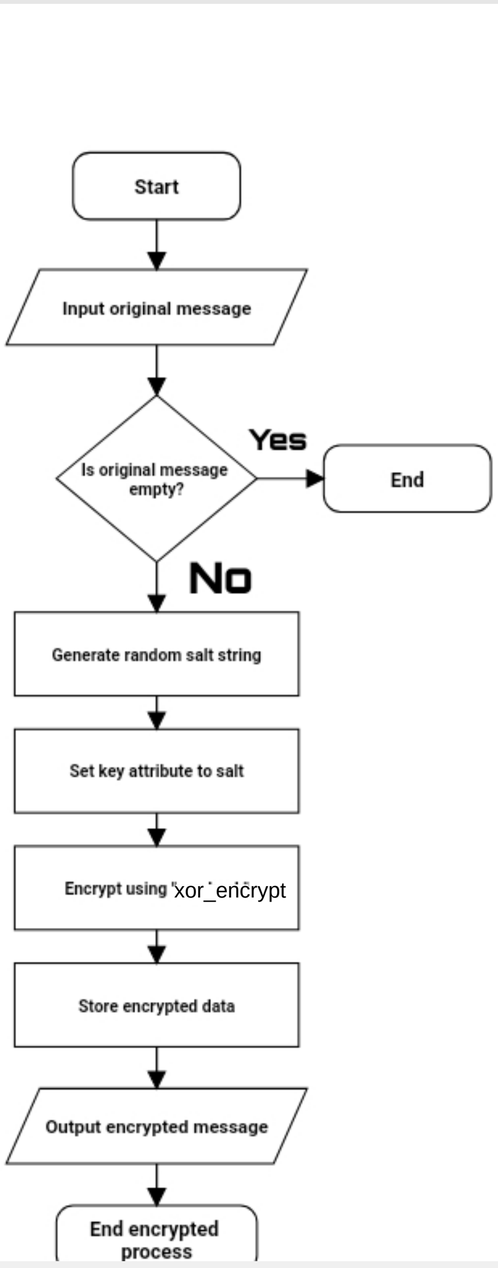
The XOR (exclusive OR) algorithm is a basic and easy-to-implement logic operation. The XOR operation is simple and it forms the basis for more complex encryption and hashing algorithms. In this algorithm, binary operation is conducted. It takes two inputs and produces an output based on the following rules: **1)** If both inputs are the same (either both 0 or both 1), the output is 0. **2)** If the inputs are different, the output is 1.
The XOR algorithm is generally used for encryption, key generation, and data integrity checking. Let's take a dive into some of the real-world problems it can solve:
Data Integrity Checking: In data transmission, XOR can be used for basic mistake detection and repair.
Signal Processing: XOR can be used to apply XOR-based changes to signals or to encode specific information in signal-processing applications like image processing or audio processing.
Hashing and Checksums: For data verification, XOR can be used to construct basic hash values or checksums. While it is insecure for cryptographic hash functions, it can nevertheless perform basic data integrity checks in non-cryptographic contexts.

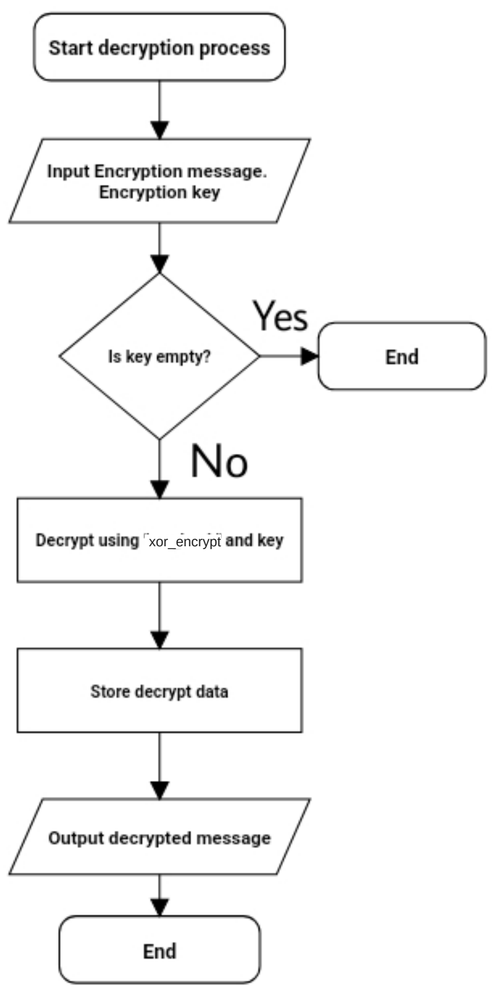
Rationale for the Algorithm's Selection

Initially, the algorithm I picked for the project was RSA algorithm. The idea behind RSA is the difficulty to factorize the product of two large prime numbers. Regardless, due to the restriction on the size of cipher text, I had to change our algorithm. I picked a simple algorithm named XOR.
It is easy to implement without using any external library. With some modifications introduced to it I managed to secure a decent score.

Methodology

This XOR-based encryption methodology is quite basic and not suitable for real-world scenarios, but it was enough for our project. Below is an overview of the code's methodology:
Initialization: The Cipher class is initialized with a constructor that takes an original message as input. If the input message is empty, the constructor returns None, indicating an invalid input.
Key Generation - Salt: The generate salt method generates a random salt of a specified length (default is 16 characters).The salt is generated using characters from the ASCII letters and digits.
XOR : The xor method takes a key and data as input and applies the XOR operation between each character of the data and the corresponding character of the key. This XOR operation is used for both encryption and decryption.
Encryption: The encrypt method converts original string to list. It sets key as salt. XOR conducts encryption using xor_encrypt and stores encrypted data in encrypted string.
Decryption: Decrypt method reverses XOR encryption using the stored key. It calls xor_encrypt to retrieve the original data. If successful, returns decrypted data.
Error Handling: The decrypt method includes error handling to catch exceptions that might occur during decryption and prints an error message if decryption fails.
Here's the flowchart of the used algorithm:





Substitute Use of the Algorithm

Substitute use of the algorithm Algorithmic substitution is the procedure of replacing one algorithm with another to produce similar or identical outcomes. This can be done for several reasons, including increasing productivity, eliminating challenges, adjusting for new limitations, or adjusting to changes in requirements. While the XOR (exclusive OR) algorithm is not as adaptable as other algorithms, it can still find applications in various domains due to its unique properties. To mention few more instances of the various applications for the XOR algorithm-
1. XOR can be used to create checksums for data transport. A checksum that is sent with the data can be created by applying XOR to a series of data bits. The received data and checksum can then be combined using XOR by the receiver to identify any problems. Checksums based on XOR are quite straightforward and can catch single-bit flaws.
2. Random number generators (RNGs) or pseudorandom number generators (PRNGs) can generate sequences of supposedly random numbers using XOR-based methods. Although their randomness and security features may not be as strong as those of more complex algorithms, using these algorithms can be quick and simple.
3. The process of concealing information within other data in a way that is difficult to find can be done with XOR. The hidden information can be made less evident by bitwise combining two pieces of data using XOR.
4. In situations where bitwise operations are advantageous, such as encoding, masking, and bitwise flipping, XOR can be used for data manipulation and transformation.
Even though XOR is not always the best option for every application, its ease of use and bitwise nature make it appropriate for some tasks where the needs match its characteristics. To provide the highest level of protection and protection against attacks, however, advanced cryptographic approaches need to be preferred for applications that are privacy critical.

Alternative Solution

I could have used AES or RSA for better overall performance or score. But due to the limitation of the size of the ciphertext, these algorithms won't be that good of a choice. I initially used RSA for this project but failed due to the restriction on the size of cipher text. Both AES and RSA with larger key sizes can result in ciphertext that is significantly larger than the original plaintext.
The scoring class is very sensitive and a minor change in the algorithm can greatly affect the overall score. The improvement made was the inclusion of random salt and it was used as a key for the xor encryption. The xor hash method takes the salt as the key parameter along with the data to be encrypted or decrypted.

Conclusion

In conclusion, the XOR algorithm, despite its simplicity, is extremely important in many fields. Even signal processing applications like picture and audio processing and data integrity verification are built on its underlying logic operation. The XOR technique is a useful tool for simple hashing, checksums, and error detection despite its drawbacks. Practical factors, such as the XOR algorithm's simplicity of implementation and the avoidance of external libraries, were the deciding factors in choosing it for a project setting. Although more complex algorithms like AES or RSA would perform better, the project's limitations, particularly the ciphertext size restriction, forced the use of XOR. The XOR algorithm serves as a reminder of the power in simplicity in a world where complicated encryption techniques are common. Its wide range of applications is demonstrated by the inclusion of it in digital logic circuits, cryptographic building blocks, and obfuscation scenarios. Understanding the benefits and drawbacks of such fundamental algorithms is still crucial for making sensible decisions in a variety of applications as technology develops.