Project 1 Report: Encryption & Decryption with Poly-Alphabetic Substitution

Contributors:
Aekansh Kathunia (2021127)
Paras Dhiman (2021482)

January 30, 2024

Abstract

This report presents the implementation and analysis of encryption, decryption, and brute-force attack techniques using poly-alphabetic substitution for secure communication. The project employs the MD5 hash function to ensure the recognition of plaintext and explores the effectiveness of a brute-force attack in discovering the encryption key. Methodology, including encryption and decryption processes, along with the algorithm for brute-force attack, are discussed. The report also provides insights into the time complexity of the brute-force approach. Through the execution of the project, successful encryption, decryption, and key discovery are demonstrated, emphasizing the importance of secure communication protocols in network security.

1 Introduction

Project 1 involves encryption and decryption using poly-alphabetic substitution. This report outlines the methodology, implementation details, and results obtained from the execution of the code.

2 Methodology:

2.1 Hash Function (MD5):

- We employed the MD5 hash function to generate a hash for the plaintext. This hash is appended to the plaintext to form recognizable text.
- The MD5 hash is converted into a string of alphabets for ease of use in our encryption and decryption processes.

2.2 Encryption:

- The encryption process involves substituting each character of the plaintext with a character from the key.
- We use a poly-alphabetic substitution approach where each character of the plaintext is shifted by the corresponding character in the key.
- The resulting ciphertext is obtained by applying this substitution for each character in the plaintext.

2.3 Decryption:

Decryption reverses the process of encryption. Each character of the ciphertext is shifted back using the key to obtain the original plaintext.

2.4 Brute Force Attack:

- Our brute-force attack method iterates through all possible combinations of a 4-character key.
- For each combination, we decrypt the ciphertext and check if the resulting plaintext satisfies the specified property.
- If a valid key is found, it is returned; otherwise, the search continues.

3 Implementation Details:

- We implemented the encryption, decryption, and brute-force attack functionalities in Python.
- The encryption and decryption functions handle both plaintext and ciphertext strings, along with the key.
- Brute force is performed by exhaustively trying all possible combinations of keys and checking each resulting plaintext.

4 Results:

4.1 Sample Input:

Original Texts:

- helloyouaredull
- whatsup
- takeachillpill
- itsreallycold
- slowandsteady

4.2 Sample Output:

Plain Texts:

- helloyouaredullbcpmmaieeecoadlplkebgmdpjjjdfedk
- whatsupfhlkcdlhimbpnhmikmelpdcfpgpeanjk
- takeachillpillcgmeemnhekfbacioepckmippldgkikma
- itsreallycoldjfgmkfioicccolckblhjfoghjhceobcl
- slowandsteadypbggjgpkehionflilnmacmjoohglgkkh

4.3 Property π Satisfaction:

- True
- True
- True
- True
- True

4.4 Encrypted Ciphertexts:

- lionscrwevhfypodgtpoemhgigrchpsnoieiqhslnnghihn
- aldvwyshlpnehpkkqfsplqlmqiorhgirkthcrnn
- xengegkkppskppfiqihorlhmjfdemshrgopkttofkolmqe
- mxvtieoncgrnhniiqoiksmfegsoeofojnjrilnkeiseep
- wpryerguxidfcteiknjroikksrinmpqoegplsskipknml

4.5 Decrypted Ciphertexts:

- helloyouaredullbcpmmaieeecoadlplkebgmdpjjjdfedk
- whatsupfhlkcdlhimbpnhmikmelpdcfpgpeanjk
- takeachillpillcgmeemnhekfbacioepckmippldgkikma
- itsreallycoldjfgmkfioicccolckblhjfoghjhceobcl
- slowandsteadypbggjgpkehionflilnmacmjoohglgkkh

4.6 Brute Force Attack Solution:

Key Chosen: "eedc" **Generated Key (Brute Force):** "eedc"

5 Brute Force Attack Analysis:

5.1 Time Complexity:

The time complexity of the brute-force attack is $O(26^4 \times n)$, where n is the number of ciphertexts to be decrypted. In this project, as the key length is fixed to 4 characters and there are 26 possible characters in each position of the key, the time complexity is $O(26^4)$ for each ciphertext. Therefore, if there are m ciphertexts to decrypt, the overall time complexity becomes $O(m \times 26^4)$.

5.2 Space Complexity:

The space complexity of the brute-force attack is $O(1)$ as it does not require any additional space proportional to the input size.

6 Conclusion:

- The project successfully demonstrates encryption, decryption, and a brute-force attack using poly-alphabetic substitution.
- The chosen hash function, MD5, efficiently facilitates the recognition of plaintext.
- By implementing the described algorithms, we achieve secure communication through encryption and reliable key discovery via brute-force attack.