CSE350: Assignment-1

Monoalphabetic Cipher

Language:- Python
Operating System:- Ubuntu

In the following paragraphs, we've provided a brief description of our system, which for the sake of simplicity, has been divided into 4 independent programs, namely, hash.py, encrypt.py, decrypt.py, and bruteforce.py.

hash.py

The program implements a hashing function using the FNV-1a algorithm. This hash function is efficient and has been chosen to avoid the avalanching effect, which was a drawback of the initially considered djb2 algorithm.

Procedure

1. **Input:** A string s
2. **Process:** Iterates over each character, applies XOR & multiplication.
3. **Output:** A 33-bit binary string representing the hash.

encrypt.py

The program converts plaintext into ciphertext using a mono-alphabetic substitution using user-defined keys.

Procedure

1. **Input Type Selection:** The user selects between binary or alphabetical input.
2. **Validation:** Ensures the input follows the required format (binary length is a multiple of 3 or valid alphabetical characters from A to H).
3. **Conversion:** Binary inputs are mapped to text using universal_set.
4. **Hash Appending:** A hash of the plaintext is appended to ensure integrity.
5. **Key Mapping:** The user provides a key to map characters A-H uniquely.
6. **Substitution:** Creates a hashmap for character substitution and encrypts the message.

decrypt.py

The program reverses the encryption process by mapping the characters back to the original text using the entered key.

Procedure

1. **Received Input:** The user inputs the encrypted text and corresponding key.
2. **Key Mapping:** A decryption map is created by reversing the key mapping.
3. **Plaintext Extraction:** Extracts the pure text and hash from the received text.
4. **Integrity Check:** Verifies the hash of the extracted text matches the received hash.

bruteforce.py

The program reverses the encryption process by mapping the characters back to the original text by checking over 8! possible keys.

Procedure

1. **Intercepted Input:** The user inputs the intercepted text.
2. **Key Mapping:** Applies each key permutation to decrypt the intercepted text.
3. **Plaintext Extraction:** Checks if the hash of the original text matches the extracted hash.
4. **Integrity Check:** If a match is found, prints the original text and elapsed time.