Project 0

CSE350: Programming Assignment 1

Parveen 2021079
Shubham Sharma 2021099

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Description

This assignment explores the implementation and cryptanalysis of a monoalphabetic substitution cipher that operates on pairs of characters.

- Encrypts plaintext messages using a substitution table where each pair of characters (**xy**, where **x** and **y** are both from the **set {A, B, C}**) is replaced with another pair of characters **(pq)**.
- Decrypts ciphertext messages back to plaintext using the same substitution table.
- Launches a brute-force attack on a given ciphertext to recover the original substitution table (i.e., the key).
- The brute-force attack will systematically test all possible substitution tables to find the one that decrypts the ciphertext into a meaningful plaintext message.

Constraints

Length of Plain Text: **512**

Hash value Length: **64**

Key Mapping used = **{**

> **'AB': 'CC', 'AC': 'BB', 'BA': 'AA', 'BC': 'CB',**
>
> **'CA': 'CA', 'CB': 'BC', 'AA': 'BA', 'BB': 'AC',**
>
> **'CC': 'AB'**

}

Plain Text generation:

- Plain text characters must be from character set {A, B, C}.
- Plain text must satisfies the property **p = (s, Hash(s))** i.e. hash of string **s** is appended in the end of string **s**.

Hashing

We use hash function to construct plaintexts that are **recognizable**,

i.e, it will satisfies the property: **p = (s, Hash(s)),** where **s** represents the plaintext and **Hash(s)** denotes its corresponding hash value.

- **Hash Function:** Utilizes the **SHA-256** hashing algorithm from the Python **hashlib** library to compute the 64-byte hash value of the input text string.

- **Custom Mapper Function:** Converts the hexadecimal hash value into a recognizable format by mapping its characters to a predefined character set **{A, B, C}**.
  - We iterate through all the characters of hash value in mapper function.
  - Then we subtracted the ASCII value of current character from ‘A’ then took it’s mod with 3 and again did its addition with ASCII value of ‘A’.
  - Returns the mapped string of hash value.

Encryption

- **Input Parameters:**

  Plain Text: Original text to be encrypted.

  Key: Mapping of character pairs to their substitutions.

Time Complexity: **O(n)**

n = length of plain text

- **Initialization:**

  Initialize an empty string for storing the cipher text.

- **Encryption Process:**

  For each pair of characters in the plaintext:

  → Find the corresponding mapping of the character pair in the provided key.
  → Append the corresponding ciphertext from the key map into the cipher text string.

- **Return:**

  Return the ciphertext string as the encrypted ciphertext.

Decryption

Time Complexity: **O(n)**
n = length of encrypted text

- **Input Parameters:**

  Cipher Text: Encrypted text to be decrypted.

  Key: Mapping of character pairs to their substitutions.

- **Initialization:**

  Initialize an empty string for storing the decrypted text.

- **Decryption Process:**

  For each pair of characters in the ciphertext:

  → Iterate through the key to find the character pair that matches the ciphertext pair.
  → Append the corresponding plaintext character pair from the key map into the decrypted text string.

- **Return:**

  Return the decrypted text string as the original plaintext.

Brute-force Attack

For the purpose of this project we have to consider the character set of **{A, B, C}** and for encryption & decryption pair of characters is taken at a time.

We can directly perform a brute-force attack by iterating through all possible combination of keys.

Verification:

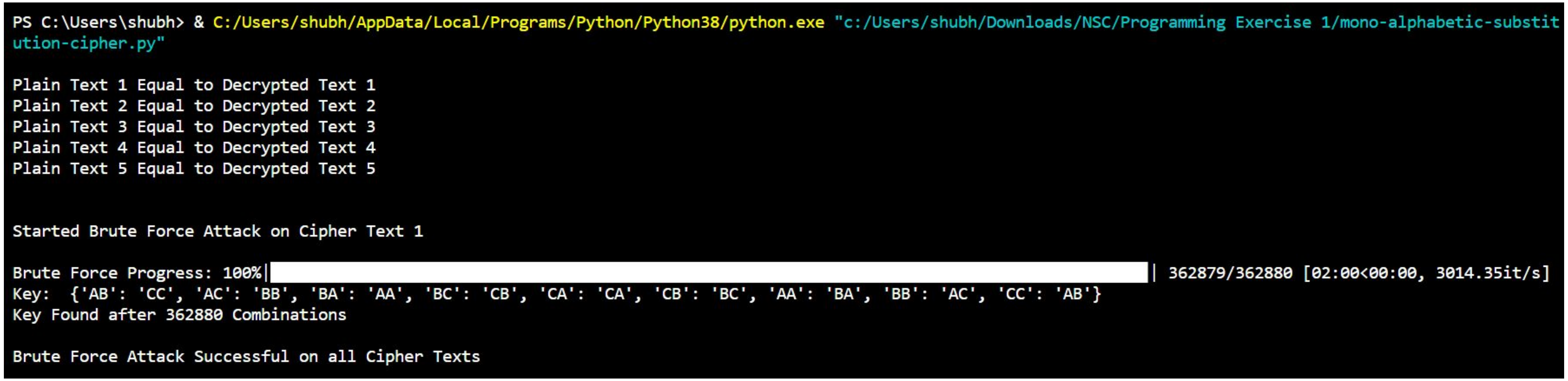
To verify whether the key generated by brute force function is correct we calculated & compared the hash value of decrypted text characters with last 64 characters of decrypted text representing the original hash value. If it matches then we return that key and decrypt rest of the four cipher texts.

Total Number of Keys: **9! → 362880**

Estimated time to successfully complete attack: **1 min 50 sec (approx)**

Asymptotic time Complexity: **O(m! * n)**, where **m** number of tuples in key and **n** is length of plain text.

Brute-force Attack Results



```
PS C:\Users\shubh> & C:/Users/shubh/AppData/Local/Programs/Python/Python38/python.exe "c:/Users/shubh/Downloads/NSC/Programming Exercise 1/mono-alphabetic-substit
ution-cipher.py"

Plain Text 1 Equal to Decrypted Text 1
Plain Text 2 Equal to Decrypted Text 2
Plain Text 3 Equal to Decrypted Text 3
Plain Text 4 Equal to Decrypted Text 4
Plain Text 5 Equal to Decrypted Text 5


Started Brute Force Attack on Cipher Text 1

Brute Force Progress: 100%|██████████████████████████████████████████████| 362879/362880 [02:00<00:00, 3014.35it/s]
Key:  {'AB': 'CC', 'AC': 'BB', 'BA': 'AA', 'BC': 'CB', 'CA': 'CA', 'CB': 'BC', 'AA': 'BA', 'BB': 'AC', 'CC': 'AB'}
Key Found after 362880 Combinations

Brute Force Attack Successful on all Cipher Texts
```