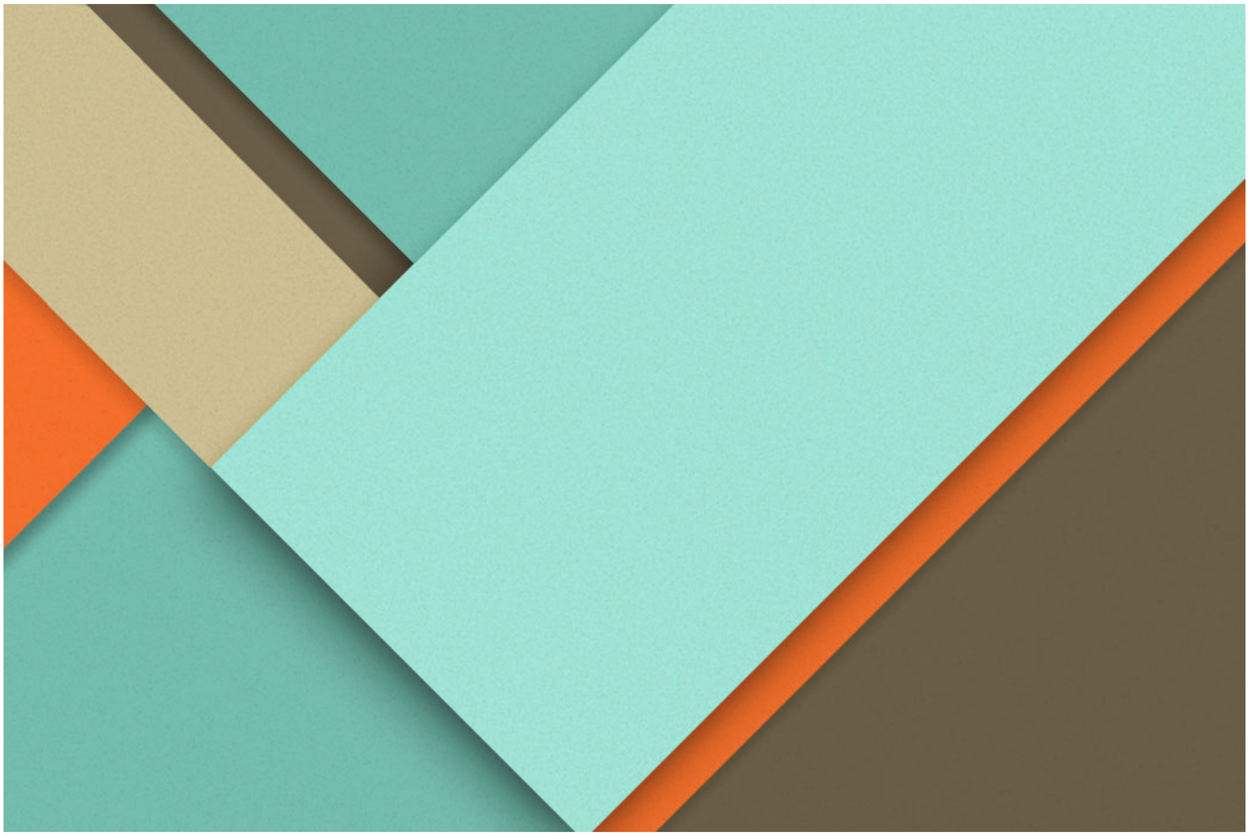


Mono - Alphabetic Substitution Cipher

Parveen	2021079
Shubham Sharma	2021099

Description

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

1. 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).
2. Decrypts ciphertext messages back to plaintext using the same substitution table.
3. Launches a brute-force attack on a given ciphertext to recover the original substitution table (i.e., the key).
4. 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:

1. Plain text characters must be from the character set **{A, B, C}**.

2. The plain text must satisfy the property **p = (s, Hash(s))**, i.e. hash of string **s** is appended at the end of string **s**.

Hashing

We use the hash function to construct plaintexts that are recognizable, i.e., it satisfies the property: **p = (s, Hash(s))**, where **s** represents the plaintext and **Hash(s)** denotes its corresponding hash value.

I. 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.

II. Custom Mapper Function

Converts the hexadecimal hash value into a recognizable format by mapping its characters to a predefined character set **{A, B, C}**.

A. We iterate through all the characters of the hash value in the mapper function.

B. Then we subtracted the ASCII value of the current character from ‘A’, took its mod with 3, and again did its addition with the ASCII value of ‘A’.

C. Returns the mapped string of the hash value.

Encryption & Decryption

Encryption

I. Input Parameters:

Plain Text: Original text to be encrypted.

Key: Mapping of character pairs to their substitutions.

II. Initialization:

Initialize an empty string for storing the cipher text.

III. Encryption Process:

For each pair of characters in the plaintext:

A. Find the corresponding mapping of the character pair in the provided key.
B. Append the corresponding ciphertext from the keymap into the cipher text string.

IV. Return:

Return the ciphertext string as the encrypted ciphertext.

Decryption

I. Input Parameters:

Cipher Text: Encrypted text to be decrypted.

Key: Mapping of character pairs to their substitutions.

II. Initialization:

Initialize an empty string for storing the decrypted text.

III. Decryption Process:

For each pair of characters in the ciphertext:

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

IV. 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, a pair of characters is taken at a time.

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

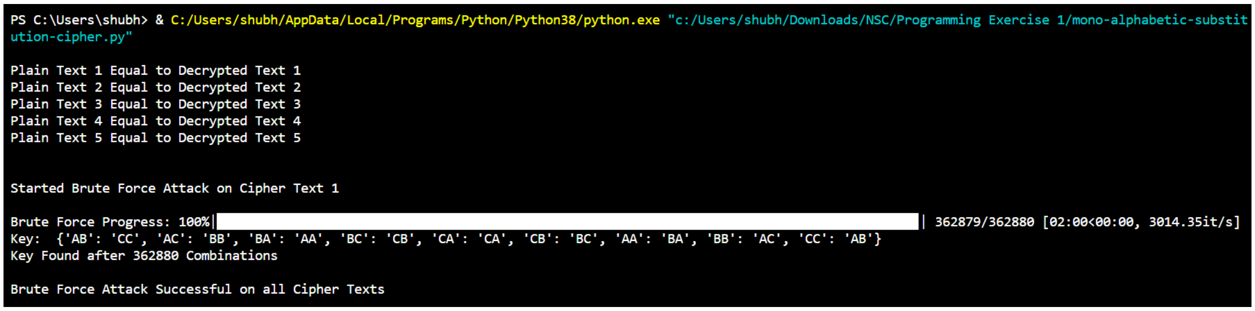
Verification

To verify whether the key generated by the brute force function is correct, we calculated & compared the hash value of decrypted text characters, with the last 64 characters of decrypted text representing the original hash value. If it matches, then we return that key and decrypt the 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** is the number of tuples in key and **n** is the length of plain text.



```
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
```

Sample Input & Output

I. Input

BCBBBCABCCCBCBBBACBBBBBAAABAABACCABBCCBCAABCBABBACACAAAAAACAB
CBACACCACABCBCBBCCABABBBBCBCABABABABBABBCCACABBCBAAAAACCBCBAB
ABACCACCBBBABACCABACACBBCAACACABABCBCBBCCCBBBAABBBBBACBCCBACC
BCBCBBABBCBACBBACACBCABCACACCAAABAAACCCBCACCBBACCCCBCBCACABAB
ACCACCABACBBCBBCBACBACBBBACAAABABCCCBBAAAACBBABCBBACCBBACBAAB
CABABAABCBCAAAABAACACCBABBAABBABACBBBCBCBCABCCAACACABCABAABBB
CCBABABCBCBBCBCBBBAAAAACBAAAABCCABCBCCCACCBCBBABBAABAACAABBCA
AAAABABBCAACABBCBBBACABABABCCABBAAACBBABBBCCCCBBABABCBBCACBBB
ABBABBBCAABBBBCAACAABAAA

Output

CBACCBCCABBCBCACBBACACAABAAACCBBCAACABCBBACBAAACBBBBBABABACAC
BAACAABBBCCBCBCCBCAAAACACBCCAAAAAAAACCCCBCACAACBCBABABBBCBCCC

CCBBCAABACAAAAABCCBBBBACCABBBBCCCCBCBCCBABACAACCACACBBCBBCBBB
CBCBCAAACBCBBACBBBBCBCCCACAABBACCBABBABCBBBBCAAABABCBCBBBCCCC
BBCAABCCBBACBCCBAABCBBACAACABAAACBABACBABABCAACBACBBBCAABCBAC
BCCCCBACBCBBABAAABBBBBCCCAACCAAAABCACBCBCCACBCABBBBCCCAAACCAC
ABAAAACBCBACBCBCACBABABBAABACCABCCBCABCAABCBACCCAACCBACACCCBB
ABACCCCCBBACAACBCACBBCCCCCCABCCAABABCAAACCBABBCAAAACBACCABCAC
CCAAACCBBAACACCABBBAAABAACABACABBCBCACACBBCAACBACAAAACCCCCCCB
BAACCCBBABCCBCCBACCACCCCCAA

II. Input

AACACBABCAACCCCABBCBCCCAACAACAAABBAAACCBABBACBAABCBAAAACBCCCC
BBBCAABACACCAAABACCCCCCCAAAACBBAAAAABABACBABABCBAABABAAACCAAA
BAACCACACAAACBBCBBCBCBACCCCBBCBBBCCBBBCCBABBBCABCCBBCBAACBBCC
AAABCCABABBACCAABABCCCCCBBCBCBCACABBCAACCACCBBCCCBBBBBCCABCCC
ABCACACBBABCCABCCAAABBABBCCABAAACCABABBBCAAAACBBCBCBABBCABBAC
ACABBCBAABCBBCBCBBBCACCCBBBCBACABBAABCCAACBCBAAACABBBBACACACC
CBCBABCCCCACCCCACBBCAACBCCBCCCCABACCBACAAACCAABCAAACABCABCCCB
BBCCAAAACCAACCAAAAAAACBCCAACCCCBBBABAABCCAAAABBCCABBCCACAACBB
ACCBCACCBBBCACBACCCBCACB

Output

BACABCCCCABBABCAACBCABCABBBACABAACBABBBCCCAABCBACBAABABBCBABB
CACCACCBBBBCABAAAABABABCABABBACBABACCCCBBAAAACBAACCCCBABBCABA
AABBCACACABABCCBACBCBCBBABBCCBACCBBCACABAAACCBCCABACBCBABCCBC
ABACBCAAAACBBCACCCCABABBCCBCBCBBBCCCBBAABBBBCCBABACACCBCACBAB
CCCACABCAACBCACBCABAACCCCBCAAABAABCCCCACCABABBACBCBCCCCBCCAAC
ACAACBCBACBACBCBCACCAABBCACBCBBCCAACCABBABCBCBABBCCACAACACAAB
BCBCCCABABBBABCABCCBBABCABCBABCAAAABAACABAABBACBBABBCCCACBABA
CCBCABABBCABBCABABABABCABBAABABACAAAACCABBABAACABCCCBCACABBAC
BBBCCAABACCBBBAAABBCCABCCAACABCBBBBBACABACCCACBCACACCBAAACBAC
ABCCBAACBCAAABCBCAACCACBBCC

III. Input

ABBAABACBAAAAACABAABBBACAACACABBAAAACBACBBBBBBCBBCCABBACACAAC
CCBCABCBBBCBCACACAAABBACCCBCBACBBCCCAACBBACBCABBCCAACACABBABC
BCACBBCACACAABBBBCCCAAACCBCCCBAABCABBCCBACABCCCBBAABAABABCBBC
BCBAACCABCAAACBABABCACBACACBBAACCABBACCACABBBBABAAAABBAAAABCA
CCBBBCAACCBAACBACCBCABABBABABACBABABCCACAAAABCBBACCCBCAACABAB
BCBAAABACAACACAAAABCCCAABAAABAACACBCBBCCBAABABCACACBCBAABCCAA

BABBBBBABCAACBABBBBACABABABACCAABACCAACBAAAABBCBAABCACACCCBBB
CABBBCBAAAABCCCCBCCCAACABBBAAAACCACCBCBACBCCCABCCBBCBBAAACCAA
BAACBCACCBBBBABBCCCBCACC

Output

CCAACCBBAABABACAAACCACBBBACACAACBABABCBBACACACBCCBCAACBBBBBAA
BBCCACBACCBCBBBBBBACCAAABBCBCBBACABCABBACBBCBCCCBCABBBBCCAACB
CBBBACCACACACCACCBABBABBBCABBCBACBCCCBBCBBCCABBCAACCBAAACBACB
CBCBAABCCCABABCCCCCCABCBBBBACBAABCCAAABBBCCACAAAABACCAABACCCA
ABACCBBAABAABBAAABCBCCCCAAAAAABCCCCCABBBBABACBACBBABCBBACAAAA
CBCBACCBBBACACABACCABCACCBACCBACABCBCCBBCBAAACBBBBBCBAACCABBA
AAACACAACBBABCCCACAACAAAAAAAABBAAAABBABCBABAACBCBACBBBBBABACC
BCCACBCBABACBABBCABCABBCCACBABAABBBBCBCBBCBABCCABACBCAABAABBA
AABBCBBBBCACAAACABBCCAABCCACCBABACAACBCBAABBBBBCCCAAACACCCCBA
BCCACCBCABBCBBCABABCCBBBABB

IV. Input

CACBCABABCBACCCBABACACCBABABBCBCAACAABBBBCCBCBBCAACBACCCBBAAB
ABCAABCCCCACACABABABBCBBACCACABCBBABABCBABACABAACABCBCCCBBBCC
ACCABAABCBAACACABABCACACBCCBACACBBACBABCBCACABBCBACABABCBCCCC
CBBBCACBBBCCCBAABBBBCAAACCBBCBCBBBCAACBCBAACBABCACCBBCBCCACAB
CACCAACCCBABBACAABCAAACBACBCACBACBBCCBCCCAAACBCACBBCBCCBBBBCB
BCACCCCACCBCABABBABAAAAAACAABBCAABACBABCACABAACBBBBCBACCCCCBA
ABBAACBCCCBABAACCACCBACBBBBBBCAABCBBACACABBBBCCBCAABBCBCAAACC
ABBBCACBBBAACACACCBCCACAABBCBABCBACBACACBBCABAAAACCCACCBABABA
AACCAACBBBCBAAABCBACACBC

Output

CABCCAAACBAAABBCCCBBBBBCCCCCCBCBBACACCACCBBCBCCBBABCBBABACBAA
ACBBACBABCACACAAAAAACBCAAABBBCCBCAAAACBAAAACAAABBCCBCABBCACAB
BBCAAACCBCBACACAAACBBBBBCBBCBBBBACBBAACBCBBBCCCBAACAAACBCBABA
BACCBBBACCBABAACCACCBBABBBCCBCBACCBBABCBCBABCCCCAABACBCABBBCC
CAABBAABBCCCAACACCCABABCBBCBBBAABCCBBCABCABABCCABCCBCBBCACCBA
CCAABABBBBCCAAAACCCBABABACACCCBBAAABCCCCACAAABBACACBCBBABABAA
CCAABBCBABAAAABBCAABAABCACACCBBACBACBBBBCCACCBBCCACCCBCBBABBC
AACCBBBACAABBBBBBBCABBBBAACBCCCBCBBAACABCCBCCBABAABCAABAAAAAA
BAABBABCACBCBACCBCBBBBCBACCCACACCAACCCBCACCBCCCCCCACAABCBBCBA
ABAABACABBACBACCCAAACBBBCAB

V. Input

```
AAABBBBBBAAACCACBCAAACCCBCABCCBAACCCABBAABBCBBACABCCABCCCACAA
BCACBBBCBAABCCCBBBBBBABCACCCBCCCABACACCACACBCABBBCCACCCACABCA
AABCACBBCABCBBCCAAACCCBCAAABABBBBABCBCACBCBBBCBBCAABCCCBAABCB
AACCAABBBBCCBACCCAACABCCACCACBCBACCBABABBCCCCBCCBACCAACCABCCA
ACCABCCAAAABBCCCBBCAAAABAACBABACBAABCCABACCAACBBBABAAACBAABAB
CACBBCCBCCCAACACACABABBACBABBAACABBCABAACCACAACABAAAACBACABAB
CACBCBACCACCAAABABBBBBCABCAABACAAACBBBABBCBCBBCAAABABBABAAACC
BCCBCBAABCAACABACBCCAABBACBBAAACCAABCAACBCABCAAAACBAAAABAAAAB
CBBABBABCCCABBBBBBCAAAAB
```

Output

```
BACCACACAABAABBBCBBABBABCBCCABAABBABCCAACCCBACBBCCABCCABCACAC
CCABCACBCBACBABACACACCCCAABBCABCAAACAABBBBBCBCCACABBBABBBCCCA
BACBBBACCACBACABBABBABCBBACCCCACAACBCBBBCBACCBACCACCABBCBACBA
ABBCACCACCBBCBBABBACACBCAABBBCBAAABAAAAACABABCBBCBBCABBCACBCA
BBCACBCABACCCBABACCABACCBABCCCBBAACCABCCBBCABBACAAAABABCBAAAC
BBBACABCBABBACACACAAAACBBAAACBACAACCAAABBCACABBCCBABABCBBCCCC
CABCBCBBCAABBACCCCACACCACBBAAACABABCACCCCBCBACCABAAAACCCBABBB
CABCBAACCCABBCCBBCBCACCAABCAABAABBACBBABCCACBBABABCBABAAABACC
BCAAACCCABCAACACACCABACCCCAAACAABBBBAACABBBCCCCBABABCCCBABCBA
CCBCAAACCACBCABABBCBBCCAAAC
```