PolyAlphabetic Cipher (Vigenère Cipher) Implementation

Arpan Kumar - 2021020

Pranav Tanwar - 2022368

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Objectives

- Implement **Vigenère Cipher** encryption & decryption.
- Use a **hash function** to validate plaintext integrity.
- Perform a **brute-force attack** to recover a 4-letter key.

Implementation Overview

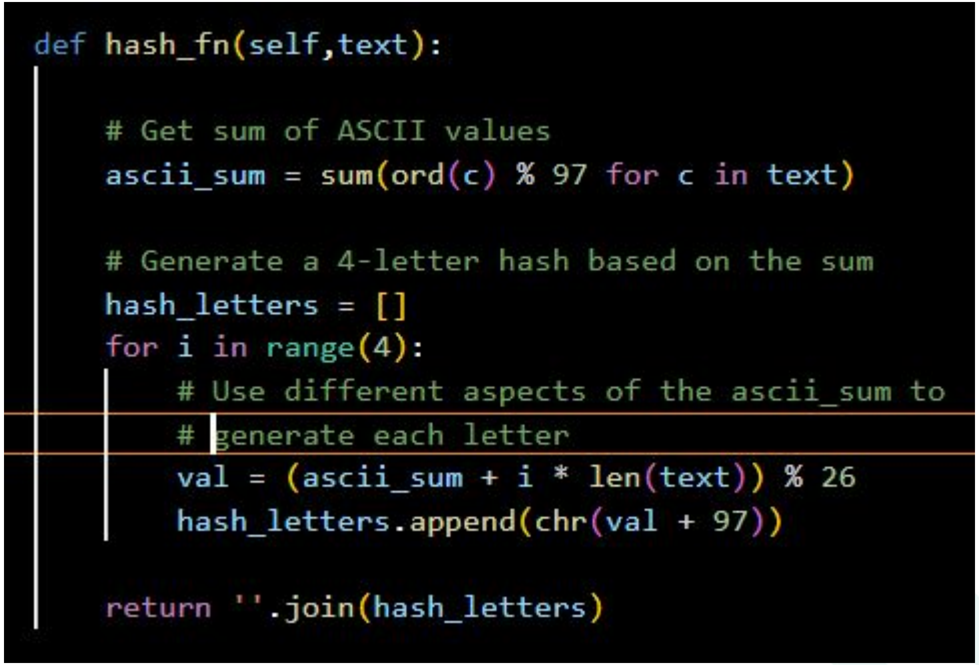
- **Classes & Functions Used:**
    a. `PolyAlphabeticCipher` → Encrypts & decrypts text.
    b. `hash_fn()` → Generates a **4-letter hash** for verification.
    c. `verify()` → Checks if the decrypted text matches the original.
    d. `brute_force()` → Attempts to **recover the key**.
- **Key Length Assumed:** 4 characters (known for brute-force).

Hash function used

Calculates ASCII value sum

Generates 4-letter hash using:

- Base ASCII sum
- Iteration index
- Text length



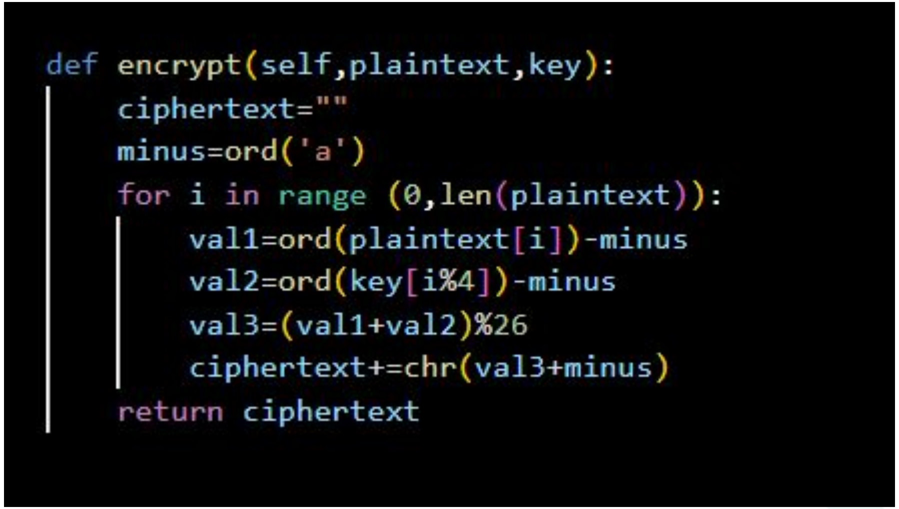
```
def hash_fn(self,text):

    # Get sum of ASCII values
    ascii_sum = sum(ord(c) % 97 for c in text)

    # Generate a 4-letter hash based on the sum
    hash_letters = []
    for i in range(4):
        # Use different aspects of the ascii_sum to
        # generate each letter
        val = (ascii_sum + i * len(text)) % 26
        hash_letters.append(chr(val + 97))

    return ''.join(hash_letters)
```

Encryption

Iterate over the plaintext and for each character do the following

Base value = ascii value of ‘a’

- Get the ascii value of the character and convert it to it’s base value by subtracting ascii value of a
- Get the ascii value of corresponding key character and convert it to the base value
- Add the above 2 values and mod it with 26 and store it in val3
- Add the character corresponding to val3+base value to the ciphertext



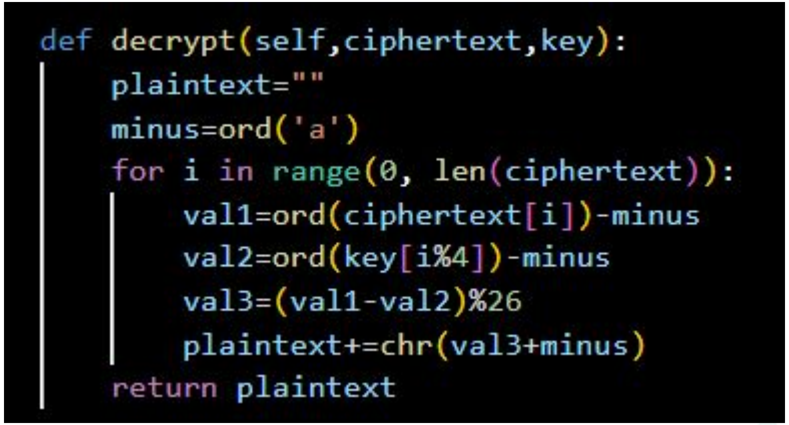
```
def encrypt(self,plaintext,key):
    ciphertext=""
    minus=ord('a')
    for i in range (0,len(plaintext)):
        val1=ord(plaintext[i])-minus
        val2=ord(key[i%4])-minus
        val3=(val1+val2)%26
        ciphertext+=chr(val3+minus)
    return ciphertext
```

Decryption

Iterate over the ciphertext and for each character do the following

Base value = ascii value of ‘a’

- Get the ascii value of the character and convert it to it’s base value by subtracting ascii value of a
- Get the ascii value of corresponding key character and convert it to the base value
- Subtract the above 2 values and mod it with 26 and store it in val3
- Add the character corresponding to val3+base value to the ciphertext



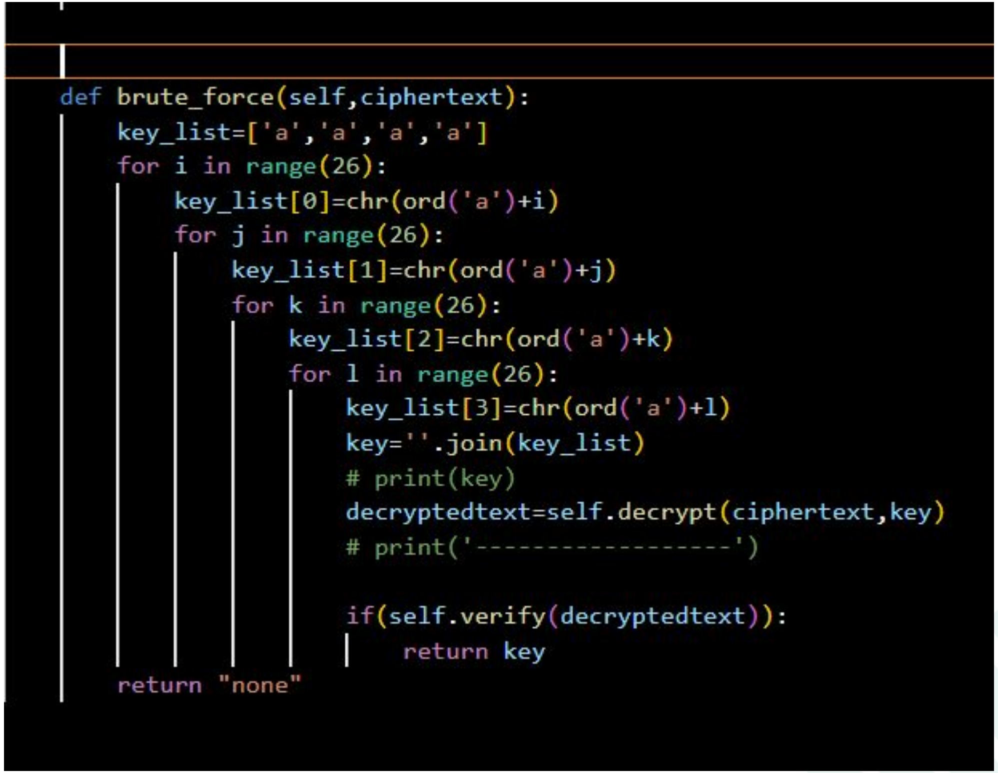
```
def decrypt(self,ciphertext,key):
    plaintext=""
    minus=ord('a')
    for i in range(0, len(ciphertext)):
        val1=ord(ciphertext[i])-minus
        val2=ord(key[i%4])-minus
        val3=(val1-val2)%26
        plaintext+=chr(val3+minus)
    return plaintext
```

Brute – force Attack

It is given that the key length is 4-bit

So, **Key Space**: 26^4=456,9762 possible keys

Trying each key to decrypt and then verifying using the **hash function**.



```
def brute_force(self,ciphertext):
    key_list=['a','a','a','a']
    for i in range(26):
        key_list[0]=chr(ord('a')+i)
        for j in range(26):
            key_list[1]=chr(ord('a')+j)
            for k in range(26):
                key_list[2]=chr(ord('a')+k)
                for l in range(26):
                    key_list[3]=chr(ord('a')+l)
                    key=''.join(key_list)
                    # print(key)
                    decryptedtext=self.decrypt(ciphertext,key)
                    # print('-----------------')

                    if(self.verify(decryptedtext)):
                        return key
    return "none"
```