Security Lab
Lab Assignment No. 5

Aim: Design and Implement Brute Force with any method.

I have implemented Brute Force with the **Caesar Cipher** method. The algorithm of Caesar cipher holds the following features:

1. Caesar Cipher Technique is the simple and easy method of encryption technique.
2. It is a simple type of substitution cipher.
3. Each letter of plain text is replaced by a letter with some fixed number of positions down with the alphabet.

The cipher text can be hacked with various possibilities. One of such possibilities is **Brute Force Technique**, which involves trying every possible decryption key. This technique does not demand much effort and is relatively simple.

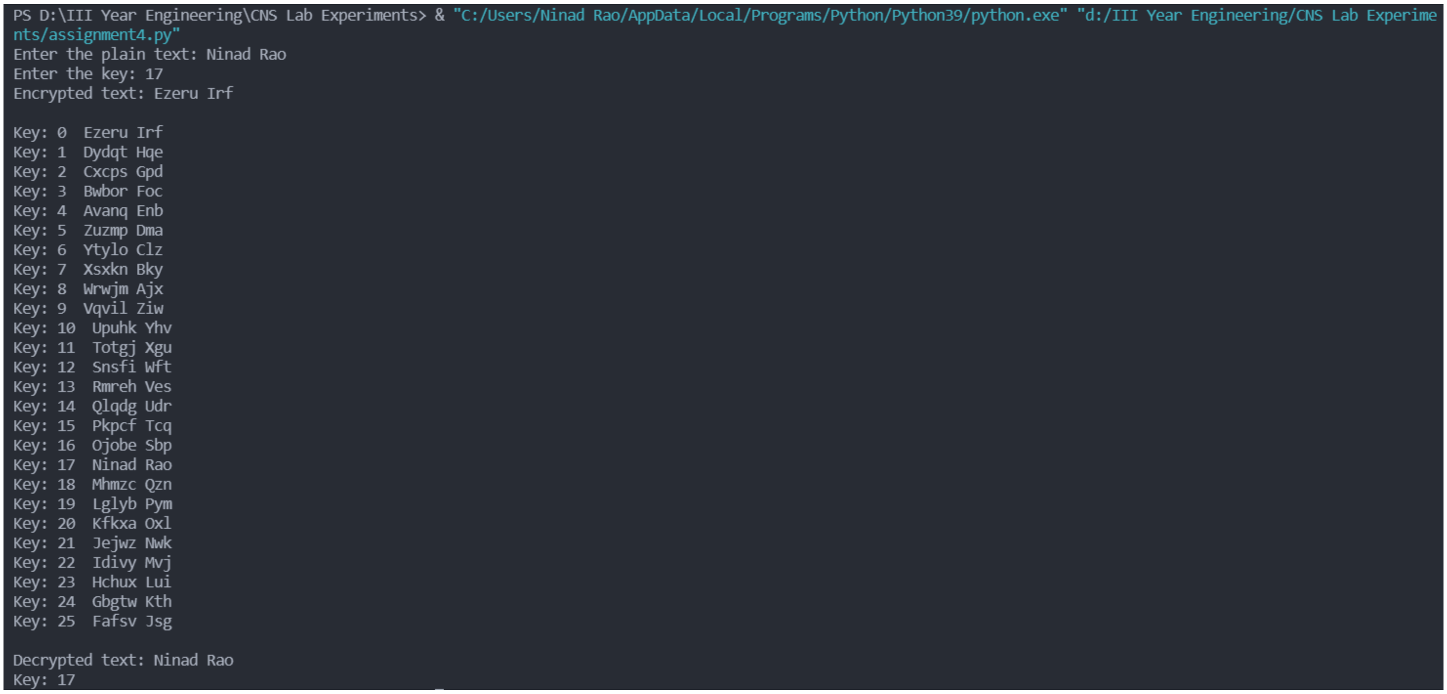
The **Brute Force Attack** method is a hit and trial method. We will analyze each of the 26 possible key combinations and try to figure out what the encrypted word is.

Code:

```
# Function to encrypt a plain text using Caesar Cipher
def encrypt(plaintext, key):
    encrypted_text = ""

    # Traverse the plain text
    for i in range(len(plaintext)):
        char1 = plaintext[i]

        # Encryption of uppercase letters
        if (char1.isupper()):
            encrypted_text += chr((ord(char1) + key - 65) % 26 + 65)

        # Keeping the whitespace as it is
        elif (char1 == ' '):
            encrypted_text += ' '
        # Encryption of lowercase letters
        else:
            encrypted_text += chr((ord(char1) + key - 97) % 26 + 97)

    return encrypted_text

# Function to decrypt an encrypted text using Caesar Cipher
def decrypt(ciphertext, plaintext):
    for i in range(26):
        decrypted_text = ''
        for j in range(len(ciphertext)):

            # Decryption of uppercase letters
            if (ciphertext[j].isupper()):
                decrypted_text += chr(65 + ((ord(ciphertext[j]) - 65 - i) % 26))

            # Keeping the whitespace as it is
            elif (ciphertext[j] == ' '):
                decrypted_text += ' '

            # Decryption of lowercase letters
            else:
                decrypted_text += chr(97 + ((ord(ciphertext[j]) - 97 - i) % 26))

        print("Key:", i, "", decrypted_text)

        if decrypted_text == plaintext:
            original = decrypted_text
            original_key = i

    print("\nDecrypted text:", original)
    print("Key:", original_key)

plaintext = input("Enter the plain text: ")
key = int(input("Enter the key: "))

# Function call for encryption
ciphertext = encrypt(plaintext, key)
```

```
print("Encrypted text:", ciphertext, "\n")

# Function call for decryption
originaltext = decrypt(ciphertext, plaintext)
```

Output:



```
PS D:\III Year Engineering\CNS Lab Experiments> & "C:/Users/Ninad Rao/AppData/Local/Programs/Python/Python39/python.exe" "d:/III Year Engineering/CNS Lab Experime
nts/assignment4.py"
Enter the plain text: Ninad Rao
Enter the key: 17
Encrypted text: Ezeru Irf

Key: 0  Ezeru Irf
Key: 1  Dydqt Hqe
Key: 2  Cxcps Gpd
Key: 3  Bwbor Foc
Key: 4  Avanq Enb
Key: 5  Zuzmp Dma
Key: 6  Ytylo Clz
Key: 7  Xsxkn Bky
Key: 8  Wrwjm Ajx
Key: 9  Vqvil Ziw
Key: 10  Upuhk Yhv
Key: 11  Totgj Xgu
Key: 12  Snsfi Wft
Key: 13  Rmreh Ves
Key: 14  Qlqdg Udr
Key: 15  Pkpcf Tcq
Key: 16  Ojobe Sbp
Key: 17  Ninad Rao
Key: 18  Mhmzc Qzn
Key: 19  Lglyb Pym
Key: 20  Kfkxa Oxl
Key: 21  Jejwz Nwk
Key: 22  Idivy Mvj
Key: 23  Hchux Lui
Key: 24  Gbgtw Kth
Key: 25  Fafsv Jsg

Decrypted text: Ninad Rao
Key: 17
```

Conclusion: Thus we understood how to design and implement Brute Force with any method.