SUB: Information Security

AY 2023-24 (Semester-V)

Jhanvi Parekh
60009210033
D11

Experiment No: 3

Aim: Design and implement Encryption and Decryption Algorithm for Caesar Cipher / Shift Cipher. Also Perform Brute Force Attack on Ciphers.

Theory:

1. Caesar Cipher / Shift Cipher.
2. Brute Force Attack on Ciphers

Example:

1) ATTACK K=3
2) ACADEMY K=25

Conclusion:

In conclusion, the Caesar cipher is a simple and historical encryption technique that involves shifting characters in a text by a fixed number of positions. While it provides basic security, it is highly vulnerable to brute force attacks due to its limited key space of 25 possible shifts (for the English alphabet). A brute force attack involves systematically trying all possible shift values to decrypt the message, making it relatively easy to break with modern computing power. To enhance security, more complex encryption methods with larger key spaces should be employed in practice.

LINK:
https://colab.research.google.com/drive/1bc6BP8wk_Qo2MH59z2RKIaaudf4wshVR?usp=sharing

Caesar Cipher

```
index = {'a': 0, 'b':1, 'c':2, 'd':3, 'e':4, 'f':5, 'g':6, 'h':7, 'i':8, 'j':9, 'k':10, 'l':11, 'm':12, 'n':13, 'o':14, 'p':15, 'q':16, 'r':1
string = input("Enter string: ")
key = int(input("Please enter a key: "))
encrypted = ''
char = list()
for x in string:
  if(index[x]==32):
    encrypted += " "
  else:
    temp = ((index[x] + key)%26)
    char += {i for i in index if index[i]==temp}

for i in range (0, len(char)):
  encrypted += char[i]
print(encrypted)
```

```
Enter string: academy
Please enter a key: 3
dfdghpb
```

```
index = {'a': 0, 'b':1, 'c':2, 'd':3, 'e':4, 'f':5, 'g':6, 'h':7, 'i':8, 'j':9, 'k':10, 'l':11, 'm':12, 'n':13, 'o':14, 'p':15, 'q':16, 'r':1
string = input("Enter string: ")
key = int(input("Please enter a key: "))
decrypted = ''
char = list()
for x in string:
  if(index[x]==32):
    encrypted += " "
  else:
    temp = ((index[x] + key)%26)
    char += {i for i in index if index[i]==temp}

for i in range (0, len(char)):
  decrypted += char[i]
print(decrypted)
```

```
Enter string: dfdghpb
Please enter a key: -3
academy
```

```
def caesar_cipher_encrypt(text):
    encrypted_text = ""
    shift = 3
    for char in text:
        if char.isalpha():
            if char.isupper():
                ascii_offset = ord('A')
            else:
                ascii_offset = ord('a')

            encrypted_char = chr(((ord(char) - ascii_offset + shift) % 26) + ascii_offset)
            encrypted_text += encrypted_char
        else:
            encrypted_text += char

    return encrypted_text

text = input("Enter the text to encrypt: ")
encrypted_text = caesar_cipher_encrypt(text)
print("Encrypted text:", encrypted_text)
```

```
Enter the text to encrypt: attack
Encrypted text: dwwdfn
```

```
def caesar_decrypt(ciphertext, shift):
    plaintext = ""
    for char in ciphertext:
        if char.isalpha():
            is_upper = char.isupper()
            char = char.lower()
            decrypted_char = chr(((ord(char) - ord('a') - shift) % 26) + ord('a'))
```

```
            if is_upper:
                decrypted_char = decrypted_char.upper()
            plaintext += decrypted_char
        else:
            plaintext += char
    return plaintext

text = input("Enter the text to decrypt: ")
shift = 3

decrypted_text = caesar_decrypt(encrypted_text, shift)
print("Decrypted Text:", decrypted_text)
```

```
 Enter the text to decrypt: dwwdfn
 Decrypted Text: attack
```

Shift cipher

```
def shift_cipher_encrypt(text, shift):
    encrypted_text = ""

    for char in text:
        if char.isalpha():
            if char.isupper():
                ascii_offset = ord('A')
            else:
                ascii_offset = ord('a')

            encrypted_char = chr(((ord(char) - ascii_offset + shift) % 26) + ascii_offset)
            encrypted_text += encrypted_char
        else:
            encrypted_text += char

    return encrypted_text

text = input("Enter the text to encrypt: ")
shift = int(input("Enter the shift value (an integer): "))

encrypted_text = shift_cipher_encrypt(text, shift)

print("Encrypted text:", encrypted_text)
```

```
 Enter the text to encrypt: Academy
 Enter the shift value (an integer): 25
 Encrypted text: Zbzcdlx
```

```
def shift_caesar_decrypt(ciphertext, shift):
    plaintext = ""
    for char in ciphertext:
        if char.isalpha():
            is_upper = char.isupper()
            char = char.lower()
            decrypted_char = chr(((ord(char) - ord('a') - shift) % 26) + ord('a'))
            if is_upper:
                decrypted_char = decrypted_char.upper()
            plaintext += decrypted_char
        else:
            plaintext += char
    return plaintext

encrypted_text = "Zbzcdlx"
shift = 25
decrypted_text = shift_caesar_decrypt(encrypted_text, shift)
print("Decrypted Text:", decrypted_text)
```

```
 Decrypted Text: Academy
```

Brute force attack

```
def BRUTE_FORCE(ciphertext, shift):
    plaintext = ""
    for char in ciphertext:
        if char.isalpha():
```

```
            is_upper = char.isupper()
            char = char.lower()
            decrypted_char = chr(((ord(char) - ord('a') - shift) % 26) + ord('a'))
            if is_upper:
                decrypted_char = decrypted_char.upper()
            plaintext += decrypted_char
        else:
            plaintext += char
    return plaintext

for i in range(0,26):
  encrypted_text = "Dfdghpb"
  shift = i
  decrypted_text = BRUTE_FORCE(encrypted_text, shift)
  print("for key : " + str(i) + " Decrypted Text: " + decrypted_text )
```

```
 for key : 0 Decrypted Text: Dfdghpb
 for key : 1 Decrypted Text: Cecfgoa
 for key : 2 Decrypted Text: Bdbefnz
 for key : 3 Decrypted Text: Academy
 for key : 4 Decrypted Text: Zbzcdlx
 for key : 5 Decrypted Text: Yaybckw
 for key : 6 Decrypted Text: Xzxabjv
 for key : 7 Decrypted Text: Wywzaiu
 for key : 8 Decrypted Text: Vxvyzht
 for key : 9 Decrypted Text: Uwuxygs
 for key : 10 Decrypted Text: Tvtwxfr
 for key : 11 Decrypted Text: Susvweq
 for key : 12 Decrypted Text: Rtruvdp
 for key : 13 Decrypted Text: Qsqtuco
 for key : 14 Decrypted Text: Prpstbn
 for key : 15 Decrypted Text: Oqorsam
 for key : 16 Decrypted Text: Npnqrzl
 for key : 17 Decrypted Text: Mompqyk
 for key : 18 Decrypted Text: Lnlopxj
 for key : 19 Decrypted Text: Kmknowi
 for key : 20 Decrypted Text: Jljmnvh
 for key : 21 Decrypted Text: Ikilmug
 for key : 22 Decrypted Text: Hjhkltf
 for key : 23 Decrypted Text: Gigjkse
 for key : 24 Decrypted Text: Fhfijrd
 for key : 25 Decrypted Text: Egehiqc
```