Dawood University of Engineering and Technology

COMPUTER AND INFORMATION SECURITY

Assignment # 01

16-Nov-2024

LAJPAT RAI | BSCS-21f-11

CIS

Sir Atif Jamil

Assignment 01

(1) mono alphabetic

a) Additive Cypher

```
def preprocess_text(text, remove_spaces=True):
    text = ''.join([c for c in text if c.isalpha()])
    if remove_spaces:
        text = text.replace(" ", "")
    return text.upper()
def additive_encrypt(plain_text, key):
    plain_text = preprocess_text(plain_text)
    cipher_text = ""
    for char in plain_text:
        shifted = (ord(char) - ord('A') + key) % 26
        encrypted_char = chr(shifted + ord('A'))
        cipher_text += encrypted_char
    return cipher_text

def additive_decrypt(cipher_text, key):
    cipher_text = preprocess_text(cipher_text)
    plain_text = ""
    for char in cipher_text:
        shifted = (ord(char) - ord('A') - key) % 26
        decrypted_char = chr(shifted + ord('A'))
        plain_text += decrypted_char
    return plain_text
plain_text = input("Enter plain text: ")
key = int(input("Enter key (0-25): "))
encrypted_text = additive_encrypt(plain_text, key)
decrypted_text = additive_decrypt(encrypted_text, key)
print("Encrypted Text:", encrypted_text)
print("Decrypted Text:", decrypted_text)
```

OUTPUT

```
e "c:/Users/PMLS/OneDrive - Dawood University of Engineering Technology/Desktop/assignmet1/Monoalp
Enter plain text: lajpat rai
Enter key (0-25): 2
Encrypted Text: NCLRCVTCK
Decrypted Text: LAJPATRAI
PS C:\Users\PMLS\OneDrive - Dawood University of Engineering Technology\Desktop\assignmet1>
```

Assignment 01

b) Multiplicative Cypher

```
def mod_inverse(a, m):
    for x in range(m):
        if (a * x) % m == 1:
            return x
    return -1
def preprocess_text(text):
    return ''.join([c.upper() for c in text if c.isalpha()])
def multiplicative_encrypt(plain_text, key):
    plain_text = preprocess_text(plain_text)
    cipher_text = ""
    for char in plain_text:
        encrypted_char = chr(((ord(char) - ord('A')) * key % 26) + ord('A'))
        cipher_text += encrypted_char
    return cipher_text
def multiplicative_decrypt(cipher_text, key):
    cipher_text = preprocess_text(cipher_text)
    inverse_key = mod_inverse(key, 26)
    if inverse_key == -1:
        raise ValueError("No modular inverse exists for the provided key.")
    plain_text = ""
    for char in cipher_text:
        decrypted_char = chr(((ord(char) - ord('A')) * inverse_key % 26) +
ord('A'))
        plain_text += decrypted_char
    return plain_text
plain_text = input("Enter plain text: ")
key = int(input("Enter key (must be coprime with 26): "))
encrypted_text = multiplicative_encrypt(plain_text, key)
decrypted_text = multiplicative_decrypt(encrypted_text, key)
print("Encrypted Text:", encrypted_text)
print("Decrypted Text:", decrypted_text)
```

OUTPUT

```
e "c:/Users/PMLS/OneDrive - Dawood University of Engineering Technology/Desktop/assignmet1/
Enter plain text: hello
Enter key (must be coprime with 26): 7
Encrypted Text: XCZZU
Decrypted Text: HELLO
PS C:\Users\PMLS\OneDrive - Dawood University of Engineering Technology\Desktop\assignmet1:
```

Assignment 01

c) Affine Cypher

```
def preprocess_text(text):
    return ''.join([c.upper() for c in text if c.isalpha()])
def mod_inverse(a, m):
    for i in range(1, m):
        if (a * i) % m == 1:
            return i
    return None
def affine_encrypt(plain_text, key1, key2):
    plain_text = preprocess_text(plain_text)
    cipher_text = ''.join([chr(((ord(c) - ord('A')) * key1 + key2) % 26 +
ord('A')) for c in plain_text])
    return cipher_text
def affine_decrypt(cipher_text, key1, key2):
    cipher_text = preprocess_text(cipher_text)
    inverse_key1 = mod_inverse(key1, 26)
    if inverse_key1 is None:
        return "Error: No modular inverse found for the key"
    plain_text = ''.join([chr((inverse_key1 * (ord(c) - ord('A') - key2)) % 26 +
ord('A')) for c in cipher_text])
    return plain_text

plain_text = input("Enter plain text: ")
key1 = int(input("Enter key1 (multiplicative key): "))
key2 = int(input("Enter key2 (additive key): "))
encrypted_text = affine_encrypt(plain_text, key1, key2)
decrypted_text = affine_decrypt(encrypted_text, key1, key2)
print(f"Encrypted Text: {encrypted_text}")
print(f"Decrypted Text: {decrypted_text}")
```

OUTPUT

```
e "c:/Users/PMLS/OneDrive - Dawood University o
Enter plain text: hello
Enter key1 (multiplicative key): 7
Enter key2 (additive key): 2
Encrypted Text: ZEBBW
Decrypted Text: HELLO
PS C:\Users\PMLS\OneDrive - Dawood University o
```

(2) poly alphabetic

a) Playfair Cypher

```
import numpy as np

# Helper function to preprocess the text
def preprocess_text(text, remove_spaces=True):
    text = text.replace(" ", "") if remove_spaces else text
    return ''.join([c.upper() for c in text if c.isalpha()])

# Helper function to create the 5x5 Playfair key matrix
def create_playfair_matrix(key):
    key = preprocess_text(key)
    key_matrix = []
    used_letters = set()

    for char in key + "ABCDEFGHIKLMNOPQRSTUVWXYZ":  # J is usually merged with I
        if char not in used_letters and char != 'J':
            key_matrix.append(char)
            used_letters.add(char)

    return np.array(key_matrix).reshape(5, 5)

# Helper function to format plaintext for Playfair cipher
def format_playfair_text(text):
    text = preprocess_text(text)
    formatted_text = ""
    i = 0
    while i < len(text):
        if i == len(text) - 1:
            formatted_text += text[i] + 'X'
            i += 1
        elif text[i] == text[i + 1]:
            formatted_text += text[i] + 'X'
            i += 1
        else:
            formatted_text += text[i] + text[i + 1]
            i += 2
    return formatted_text

# Function to find position of a letter in the matrix
def find_position(matrix, letter):
```

Assignment 01

```
    for row in range(5):
        for col in range(5):
            if matrix[row][col] == letter:
                return row, col
    return None, None

# Playfair encryption
def playfair_encrypt(plain_text, key):
    matrix = create_playfair_matrix(key)
    plain_text = format_playfair_text(plain_text)
    cipher_text = ""

    for i in range(0, len(plain_text), 2):
        r1, c1 = find_position(matrix, plain_text[i])
        r2, c2 = find_position(matrix, plain_text[i + 1])

        if r1 == r2:
            cipher_text += matrix[r1][(c1 + 1) % 5] + matrix[r2][(c2 + 1) % 5]
        elif c1 == c2:
            cipher_text += matrix[(r1 + 1) % 5][c1] + matrix[(r2 + 1) % 5][c2]
        else:
            cipher_text += matrix[r1][c2] + matrix[r2][c1]

    return cipher_text

# Playfair decryption
def playfair_decrypt(cipher_text, key):
    matrix = create_playfair_matrix(key)
    plain_text = ""

    for i in range(0, len(cipher_text), 2):
        r1, c1 = find_position(matrix, cipher_text[i])
        r2, c2 = find_position(matrix, cipher_text[i + 1])

        if r1 == r2:
            plain_text += matrix[r1][(c1 - 1) % 5] + matrix[r2][(c2 - 1) % 5]
        elif c1 == c2:
            plain_text += matrix[(r1 - 1) % 5][c1] + matrix[(r2 - 1) % 5][c2]
        else:
            plain_text += matrix[r1][c2] + matrix[r2][c1]

    return plain_text

# Input and Output demonstration
if __name__ == "__main__":
```

Assignment 01

```
plain_text = input("Enter the plain text: ")
key = input("Enter the key: ")

encrypted_text = playfair_encrypt(plain_text, key)
print("Encrypted Text:", encrypted_text)

decrypted_text = playfair_decrypt(encrypted_text, key)
print("Decrypted Text:", decrypted_text)
```

OUTPUT

```
e "c:/Users/PMLS/OneDrive - Dawood Universi
Enter the plain text: move
Enter the key: MONARCHY
Encrypted Text: ONUF
Decrypted Text: MOVE
PS C:\Users\PMLS\OneDrive - Dawood Universi
```

b) Autokey Cypher

```
def preprocess_text(text):
    return ''.join([c.upper() for c in text if c.isalpha()])
def autokey_encrypt(plain_text, key):
    plain_text = preprocess_text(plain_text)
    key_stream = (key + plain_text)[:len(plain_text)]
    cipher_text = ""
    for p, k in zip(plain_text, key_stream):
        encrypted_char = chr((ord(p) - ord('A') + ord(k) - ord('A')) % 26 +
ord('A'))
        cipher_text += encrypted_char
    return cipher_text
def autokey_decrypt(cipher_text, key):
    cipher_text = preprocess_text(cipher_text)
    key_stream = key
    plain_text = ""
    for i in range(len(cipher_text)):
        decrypted_char = chr((ord(cipher_text[i]) - ord('A') -
(ord(key_stream[i]) - ord('A'))) % 26 + ord('A'))
        plain_text += decrypted_char
        key_stream += decrypted_char
    return plain_text
plain_text = input("Enter plain text: ")
key = input("Enter key: ")
encrypted_text = autokey_encrypt(plain_text, key)
decrypted_text = autokey_decrypt(encrypted_text, key)

print("Encrypted Text:", encrypted_text)
print("Decrypted Text:", decrypted_text)
```

OUTPUT

```
e "c:/Users/PMLS/OneDrive - Dawood University o
Enter plain text: attack
Enter key: M
Encrypted Text: MTMTCM
Decrypted Text: ATTACK
PS C:\Users\PMLS\OneDrive - Dawood University o
```

c) Vigenère Cypher

```
def preprocess_text(text):
    return ''.join([c.upper() for c in text if c.isalpha()])

def vigenere_encrypt(plain_text, key):
    plain_text = preprocess_text(plain_text)
    key = (key * ((len(plain_text) // len(key)) + 1))[:len(plain_text)]
    cipher_text = ''.join([chr((ord(p) - ord('A') + ord(k) - ord('A')) % 26 +
ord('A')) for p, k in zip(plain_text, key)])
    return cipher_text

def vigenere_decrypt(cipher_text, key):
    cipher_text = preprocess_text(cipher_text)
    key = (key * ((len(cipher_text) // len(key)) + 1))[:len(cipher_text)]
    plain_text = ''.join([chr((ord(c) - ord('A') - (ord(k) - ord('A'))) % 26 +
ord('A')) for c, k in zip(cipher_text, key)])
    return plain_text

plain_text = input("Enter plain text: ")
key = input("Enter key: ")
encrypted_text = vigenere_encrypt(plain_text, key)
decrypted_text = vigenere_decrypt(encrypted_text, key)
print(f"Encrypted Text: {encrypted_text}")
print(f"Decrypted Text: {decrypted_text}")
```

OUTPUT

```
e "c:/Users/PMLS/OneDrive - Dawood University of Enginee
Enter plain text: she is listening music
Enter key: PASCAL
Encrypted Text: HHWKSWXSLGNTCGEWSTR
Decrypted Text: SHEISLISTENINGMUSIC
PS C:\Users\PMLS\OneDrive - Dawood University of Enginee
```