Project 1: Encryption/Decryption Scheme

Network and Information Security

CT-486

Submitted by:

Fatima Shehzad

CT-20036

Section A

Circular XOR Cipher:

I've developed a unique scheme called Circular XOR, inspired by a close examination of the Vigenere cipher's substitution technique. Circular XOR involves performing the XOR operation followed by a modulo 26 operation. This innovative approach combines elements from various cryptographic methods to create a novel encryption technique.

Design:

1. Input:
   - Plain text provided by the user.
   - Keyword used for generating the key (in this case, considering 'FATIMA').

2. Key Generation:
   - Repeat the keyword in a circular manner to match the length of the plaintext.
   - For example, if the plaintext is 'GEEKFORGEEKS' and the keyword is 'FATIMA', the key becomes 'FATIMAFATIMAF'.

3. Conversion:
   - Convert the characters A-Z into numbers 0-25.
   - Pair each letter of the plaintext with the corresponding letter of the key for encryption.

4. Encryption:
   - Perform XOR operation between the numerical values of the plaintext and key.
   - Apply modulo 26 to the result to obtain the encrypted numerical value.
   - Convert the encrypted numerical value back to the corresponding letter.

Explanation:

The encryption scheme utilizes the XOR operation between the plaintext and the key generated from the keyword. XORing the numerical values of the characters ensures that each letter in the plaintext is mixed with a corresponding letter from the key in a reversible manner.
The modulo operation (mod 26) is applied to ensure the encrypted numerical value remains within the range of 0-25, representing the letters A-Z in the English alphabet.

Analysis:

1. Time Complexity:
 - Key Generation: O(n), where n is the length of the plaintext.
 - Encryption Process: O(n), where n is the length of the plaintext.
 - Overall, the time complexity is linear with respect to the length of the plaintext.

2. Space Complexity:
 - The space complexity is also O(n) as additional memory is required to store the key and the encrypted text.

3. Security Consideration:
 - XOR-based encryption, as described, is susceptible to known-plaintext attacks if the same key is reused for multiple messages. For increased security, using a unique key for each encryption is recommended.

4. Efficiency:
 - XOR operations are computationally efficient and fast.

Implementation:

Encryption

```
def encrypt(plaintext, keyword):
    plaintext = plaintext.upper().replace(" ", "")
    keyword = keyword.upper()
    key = ""

    # Generating the key by repeating the keyword in a circular manner
    for i in range(len(plaintext)):
        key += keyword[i % len(keyword)]

    encrypted_text = ""

    for i in range(len(plaintext)):
        # Converting A-Z to numbers 0-25
```

```
        plain_num = ord(plaintext[i]) - ord('A')
        key_num = ord(key[i]) - ord('A')

        # XOR operation and modulo 26
        encrypted_num = (plain_num ^ key_num) % 26

        # Converting back to a letter
        encrypted_text += chr(encrypted_num + ord('A'))

    return encrypted_text

# Example usage
plaintext = input("Enter the plaintext: ")
keyword = 'FATIMA'

encrypted_text = encrypt(plaintext, keyword)
print("Encrypted text:", encrypted_text)
```

Decryption

```
def decrypt(encrypted_text, keyword):
    encrypted_text = encrypted_text.upper().replace(" ", "")
    keyword = keyword.upper()
    key = ""

    # Generating the key by repeating the keyword in a circular manner
    for i in range(len(encrypted_text)):
        key += keyword[i % len(keyword)]

    decrypted_text = ""

    for i in range(len(encrypted_text)):
        # Converting A-Z to numbers 0-25
        encrypted_num = ord(encrypted_text[i]) - ord('A')
```

```
        key_num = ord(key[i]) - ord('A')

        # XOR operation and modulo 26 to decrypt
        decrypted_num = (encrypted_num ^ key_num) % 26

        # Converting back to a letter
        decrypted_text += chr(decrypted_num + ord('A'))

    return decrypted_text

# Example usage
encrypted_text = input("Enter the encrypted text: ")
keyword = 'FATIMA'

decrypted_text = decrypt(encrypted_text, keyword)
print("Decrypted text:", decrypted_text)
```

Testing:

Sample 1:

Encryption:

Plain Text: GEEKFORGEEKS

Key: FATIMA

Keyword: FATIMAFATIMAF

Cipher Text: DEXCTOUGXMGS

Decryption:

Cipher Text: DEXCTOUGXMGS

Plain Text: GEEKFORGEEKS

Sample 2:

Encryption:

Plain text: WELCOME

Key: FATIMA

Keyword: FATIMAF

Cipher text: TEYKCMB

Decryption:

Encrypted text/Cipher text: TEYKCMB

Decrypted text/Plain text : WELCOME