**Lee\_Report\_PA3**

**Cryptography Section W01 Summer Semester 2024 CO**

**Raehyeong Lee**

**2024-07-06**

**Overview**

This project implements a single round of the Feistel cipher, a symmetric structure used in many block ciphers, to encrypt text. The Feistel cipher round function takes an 8-bit binary input and a 4-bit key, divides the input into two halves, and processes them using a specified function and XOR operation. The project also includes a text converter to handle string inputs, converting each character to its binary representation, applying the Feistel round function, and converting the result back to text. This approach demonstrates how a simple cryptographic function can be applied iteratively to encrypt text.

**Functions**

**1. feistel\_round(input\_bits, key)**

This function implements a single round of the Feistel cipher.

* Takes an 8-bit binary string and a 4-bit key as input.
* Splits the input into two 4-bit halves, 𝐿0 and R0.
* Computes the round function F(R0,k) =(2)
* Returns the concatenated result of and .

**2. text\_to\_binary(text)**

This function converts a text string to its binary representation.

* Iterates over each character in the text.
* Converts each character to its 8-bit binary representation.
* Returns the concatenated binary string.

**3. binary\_to\_text(binary\_str)**

This function converts a binary string back to text.

* Splits the binary string into 8-bit chunks.
* Converts each chunk to its corresponding character.
* Returns the concatenated text string.

**4. feistel\_cipher\_text(text, key)**

This function applies the Feistel cipher round function to each character in the input text.

* Converts the input text to its binary representation.
* Processes each 8-bit chunk with the feistel\_round function.
* Converts the result back to text.

**Conclusion**

This project successfully implements a single round of the Feistel cipher, demonstrating its application to encrypt text. The functions ensure the correct processing of binary data, while the text converter allows for easy handling of string inputs. The resulting ciphertext shows the effectiveness of the Feistel structure in encryption. Further development could involve implementing multiple rounds for increased security and exploring different round functions.

**Project Code**

def feistel\_round(bits, k):

    assert len(bits) == 8

    assert len(k) == 4

    L = int(bits[:4], 2)

    R = int(bits[4:], 2)

    key = int(k, 2)

    Rwithk = (2 \* (R \*\* k)) % 16

    L1 = R

    R1 = L ^ Rwithk

    L1binary = format(L1, '04b')

    R1binary = format(R1, '04b')

    return R1binary + L1binary

def strtobin(t):

    return ''.join(format(ord(c), '08b') for c in t)

def bintostr(binstr):

    char = [chr(int(binstr[i:i+8], 2)) for i in range(0, len(binstr), 8)]

    return ''.join(char)

def feistelcipher(t, k):

    bin = strtobin(t)

    result = ""

    for i in range(0, len(bin), 8):

        bit = bin[i:i+8]

        result += feistel\_round(bit, k)

    return bintostr(result)

plaintext = input("String: ")

key = input("4-bit Key: ")

ciphertext = feistelcipher(plaintext, key)

print("Ciphertext:", ciphertext)

**Output**

**텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**