TC-assignment-2: Exhaustive Search and TMTO Attack on a TOY CIPHER(TC)

Due date - 28 January 2025

You are given a toy cipher with basic encryption mechanisms that operates on 32-bit plaintexts and produces 32-bit ciphertexts using a 32-bit key. This assignment has three parts:

1. Implement the Toy Cipher (TC).
2. Implement an exhaustive search to recover the key given a plaintext-ciphertext pair.
3. Implement the Time-Memory Trade-Off (TMTO) attack to recover the key given a plaintext and the corresponding ciphertext.

Submit your code in a well-documented file, clearly indicating: any dependencies and libraries used, and the instructions (if any) for running your code on an online compiler.

Description of the Toy Cipher (TC)

The toy cipher has two main components:

1. **Round Key:** For the sake of simplicity we assume that the **same key** is used in all the rounds. Choose a random 32 bit string as a key and split the 32-bit string into **four** 8-bit strings.

   For Example - the 32 bit string $0x9ABCDEF0$ would be split as $0x9A, 0xBC, 0xDE, 0xF0$

2. **Encryption Function:** Implements the encryption process using the keys chosen above.

The encryption function is described below.

```
# TC1 - Toy Cipher (encryption)
# - Parameters:
#   - n, k: 32-bit inputs (4 cells of 1 byte each)
#   - NUM_ROUND: Number of encryption rounds (fixed to 10)
#   - Sbox: AES Sbox

# Add Roundkey (AR)
def AR(in_state, rkey):
    out_state = [0] * len(in_state)
```

```
    for i in range(len(in_state)):
        out_state[i] = in_state[i] ^ rkey[i]
    return out_state

# S-Box Layer (SB)
def SB(in_state):
    out_state = [0] * len(in_state)
    for i in range(len(in_state)):
        out_state[i] = Sbox[in_state[i]]
    return out_state

# Linear Mixing (LM)
def LM(in_state):
    out_state = [0] * len(in_state)
    All_Xor = in_state[0] ^ in_state[1] ^ in_state[2] ^
       in_state[3]
    for i in range(len(in_state)):
        out_state[i] = All_Xor ^ in_state[i]
    return out_state

# Single Encryption Round
def Enc_Round(in_state, rkey):
    out_state = AR(in_state, rkey)     # Add Roundkey
    in_state = SB(out_state)          # Apply S-box layer
    out_state = LM(in_state)          # Apply Linear Mixing
    return out_state

# TC1 Encryption
def TC1_Enc(PT, key):
    NROUND = NUM_ROUND  # Number of rounds
    CT = PT             # Initialize ciphertext with plaintext
    for i in range(NROUND):
        CT = Enc_Round(CT, key)  # Perform encryption round
    return CT
```

The Assignment

Part 1: Implementation of the Toy Cipher (TC)

Implement the above toy cipher. For test case consider the following:

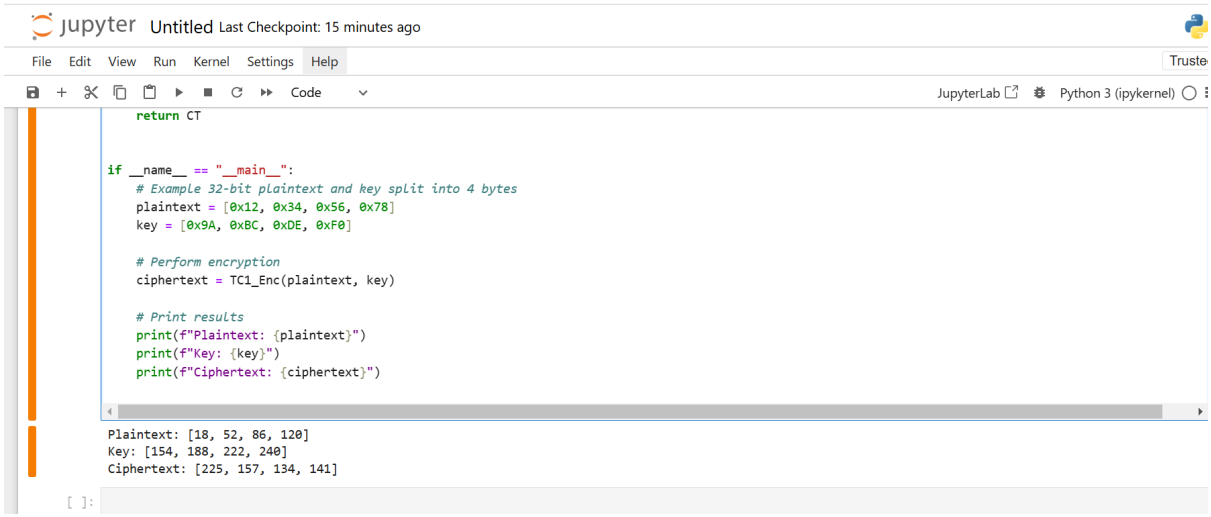

Figure 1: Toy Cipher (TC)

Part 2: Exhaustively Searchig for the key

Consider the following pairs of plaintext-ciphertext encrypted by TC using the same key k. Find the key k exhaustively.

1. $p = [0, 0, 0, 0]$ and $c = [180, 31, 145, 240]$
2. $p = [1, 2, 3, 4]$ and $c = [228, 99, 105, 79]$

Note: Conisdering that 2^{32} would take a lot of time in your machine, consider the first byte of the key to be $0x00$, i.e the key is of the form $k = [0x00, ?, ?, ?]$. So, you only have to search for the remaining 24 bits of the key

Part 3: TMTO Attack for key recovery

Consider that you are provided with a plaintext $p = [18, 52, 86, 120]$ and its corresponding ciphertext $c = [86, 116, 35, 75]$. Then search for the key using TMTO attack. For this, you will need to do the following:

I. Construct the pre-processing Table T

   Generate m random 32 bit strings

   construct the chains as described in the class

   Store the first and last element of the chain as SP and EP

II. Search for the element x in the table such that $TC(p, x) = c$.

Note: you can perform this attack with your own set of plaintext and ciphertext also. Hereyou have to search for the full 32 bit key.