Report Lab 1

Group: sudo rm -RF

**2.1 Implementation**

* **Task 1**: we have implemented each part of the AES-like cipher as a isolated method, the encryption function is built in a manner that it’s flexible to various different functions applied to it.
* **Task 2**: As for the encryption, each part of the decryption is implemented as an isolated function. Then the decryption function take in input as a parameter a function for each stage of the decryption (sub-key generator, sub-key sum, substitution, transposition and linear). The result is a modular function that can adapt and supply a decryption function for different type of encryption.
* **Task 3**: we have identified both the matrix A and B using the black box approach, implemented in the function: **linear\_relationship**
* **Task 4**: we took the plaintext-ciphertext pairs, computed the formula to find the key of the pairs
* **Task 5**: a switch like method, each value is changed in accordance to the table
* **Task 6**: we construct a linear approximation of the non-linear substitution function called approx\_sub and used as substitution function in the encryptor; so we used the function linear\_relationship (task 3) on this encryption function in order to find the matrices A and B. Then evaluated the linear approximation on a generated dataset of 50000 samples. Finally we tried to found the key for the 5 pairs in the file using first the linear approximation to find a candidate key and then searching keys close to it. Since we got 5 different keys and we are searching for one that is a neighbor for each of them we construct an aggregate key of the found 5 and then searched in the neighbor of it.
* **Task 7**: we implemented the new sub-key generation in a new method, with the new substitution function and passed in the encryption method
* **Task 8**: For the “meet in the middle” attack we implemented:

- double\_enc function made of two encryption function of task 7  
- decryption function for the encryption function of task 7   
- a function to generate the set of keys  
- a forward step function that takes the plaintext u, encrypt it using the task 7 encryption function and each of the keys generated, and add the tuple (k,x) in a list. At the end sorts the list by the x and return it

- a backward step function that takes cipher text x, decrypt it using the corresponding decryption function and each of the keys generated, and add the tuple (k,u) in a list. At the end sorts the list by the u and return it  
- the meet in the middle function that call the preceding two function to generate the two sorted lists (k1,x1) and (k2,u2) and then search for all matches x1 == u2 since more pairs of keys can give the same cipher and we need to find the only pair of keys that encrypt the 5 pairs plaintext-ciphertext in the file.

**2.2 Results:**

**2.2.1 Matrices task 3**

A = [[9, 0, 1, 6, 0, 0, 1, 10],

[0, 8, 6, 2, 2, 9, 0, 0],

[0, 6, 0, 8, 3, 10, 0, 0],

[6, 0, 0, 8, 0, 1, 6, 6],

[2, 0, 1, 10, 0, 0, 1, 3],

[0, 1, 8, 4, 9, 6, 0, 0],

[0, 10, 0, 5, 7, 6, 0, 0],

[3, 0, 0, 1, 0, 1, 4, 8]]

B = [[6, 0, 0, 3, 3, 0, 0, 0],

[0, 6, 3, 0, 0, 3, 0, 0],

[0, 3, 6, 0, 0, 0, 3, 0],

[3, 0, 0, 6, 0, 0, 0, 3],

[5, 0, 0, 0, 4, 0, 0, 8],

[0, 5, 0, 0, 0, 4, 8, 0],

[0, 0, 5, 0, 0, 8, 4, 0],

[0, 0, 0, 5, 8, 0, 0, 4]]

**2.2.2 Linear KPA guess**

k = [9, 1, 4, 3, 10, 6, 2, 1]

**2.2.3 Matrices task 6**

A = [[ 9, 0, 1, 6, 0, 0, 1, 10]

[ 0, 8, 6, 2, 2, 9, 0, 0]

[ 0, 6, 0, 8, 3, 10, 0, 0]

[ 6, 0, 0, 8, 0, 1, 6, 6]

[ 2, 0, 1, 10, 0, 0, 1, 3]

[ 0, 1, 8, 4, 9, 6, 0, 0]

[ 0, 10, 0, 5, 7, 6, 0, 0]

[ 3, 0, 0, 1, 0, 1, 4, 8]]

B = [[6, 0, 0, 3, 3, 0, 0, 0]

[0, 6, 3, 0, 0, 3, 0, 0]

[0, 3, 6, 0, 0, 0, 3, 0]

[3, 0, 0, 6, 0, 0, 0, 3]

[5, 0, 0, 0, 4, 0, 0, 8]

[0, 5, 0, 0, 0, 4, 8, 0]

[0, 0, 5, 0, 0, 8, 4, 0]

[0, 0, 0, 5, 8, 0, 0, 4]]

C = I

P[Ak + Bu + Cx = 0 mod p] = 0.000340

**2.2.4 Nearly linear KPA guess**

k =[]

**2.2.5 MITM guess**

k’ = [0, 7, 2, 4] k’’= [ 2, 9, 10, 7]