INFSCI 2170

Xiaojun Lei (xil222)

2019/4/14

## Final Project Report

#### Introduction and Purpose

DES is a symmetric encryption algorithm that was being used as a standard from 1976 to 2002. In order to understand the encryption of DES better, to test the runtime of its encryption and decryption in different modes, I made an implementation of DES using Python 3.6.

#### Files

There are 8 python files in the folder:

1. Table.py stores all permutation tables and substitution tables;
2. Convert.py stores helper functions that could transform binary strings to hexadecimal strings, plaintext strings to hexadecimal strings, and etc;
3. Key.py stores the Key class, which transforms the 64-bit key input to a key stream;
4. Des.py is the class of DES and includes the feistel function and the encryption process of DES;
5. TriDes.py is the class of Triple DES
6. Main.py is the first main method that simply test the encryption and decryption of DES, and
7. Main2.py is used to test the runtime on different modes without multiprocessing;
8. Main3.py is the runtime test file for multiprocessing

In order to make the input easier to be processed, all binary and hexadecimal inputs are string-type variables.

#### Implementation

I implemented four different type of DES systems: the original mode, ECB mode, CBC mode, and Triple DES. The original mode only takes a 64-bit binary string as an input; ECB mode and CBC mode will break the input string into a list of 64-bit binary strings, with padding “0”s in the last block to make it a multiple of 64 bits. ECB mode encrypts and decrypts each block using same key and transmit, while CBC mode will take an xor operation before encrypting the block. Triple DES concatenates 3 original DES. To test the DES function, data from different homework assignments and websites are used, and check the result of encryption by decrypting the ciphertext. Encrypting text messages is also possible.

#### Runtime

Without multiprocessing, ECB and CBC take about 30 seconds to encrypt a text with about 160 words 200 times, and Triple DES takes 90 seconds to complete the same task, which is 3 times of the normal DES. I attempted to add a multiprocessing on the normal DES module, but it seems that the multiprocessing will take much more time. Maybe it is due to my wrong implement method.

#### Other Findings

There is one interesting founding when I was trying to debug the code: when one digit in the permutation table or substitution table is messed up, only specific digits will have error, while others containing correct information.

## Reference

1. <http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm>
2. Stallings, W. *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 1999.
3. Slides from CourseWeb