

COMP 9337 Securing Wireless Networks

T1, 2019

Lab 1

Group: SWN19 AI

Zhou JIANG (z5146092), Wanze LIU (z5137189)

School of Computer Science and Engineering

UNSW Sydney

README

This code should be compatible for **Python3.6**, which has been tested on CSE machine (module **matplotlib** already installed in CSE machine for Python3.6). Please typing:

```
python3 <python file > <arguments...>
```

to run the codes.

HOW TO TEST

1. For simple test, just type:

```
python3 run.py
```

to generate all test files in different size and automatically test them, drawing bitmaps of time consumption.

2. If you want to test **tempdes.py** file ONLY, please refer to the lab description for command-line-arguments.
3. **temppaes.py** takes two argument: <input file> <output file>.
4. **Other python codes** each takes one argument: <input file>.

IMPLEMENTATION

Python code implementation in parts 3, 4.c and 4.d

In part 3.

1. We encrypt file by using the key and IV that provided by the lab1 materials (README)
2. After encryption finished, we write the encrypted message to the file named 'mytest.des'.
3. Checking whether the encryption file is matching the 'test.des' file provided.
4. Using checksum method to see if it matchs by typing command ' md5sum file name ' in the shell interface.

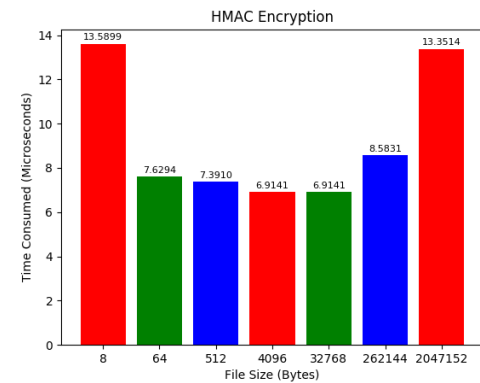
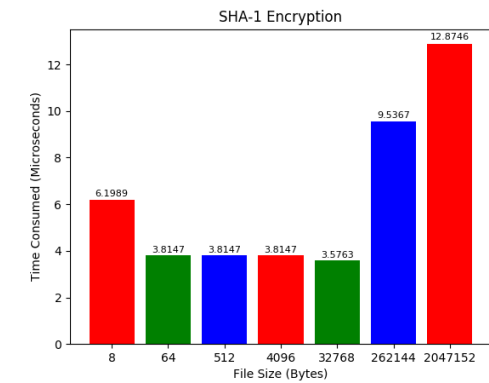
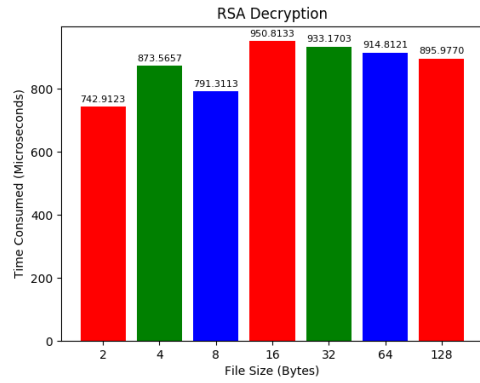
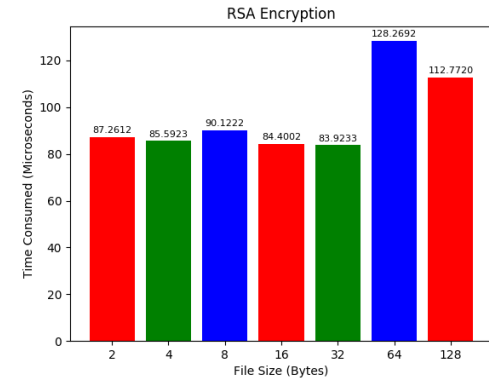
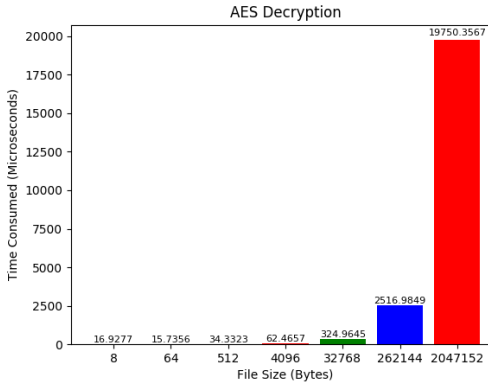
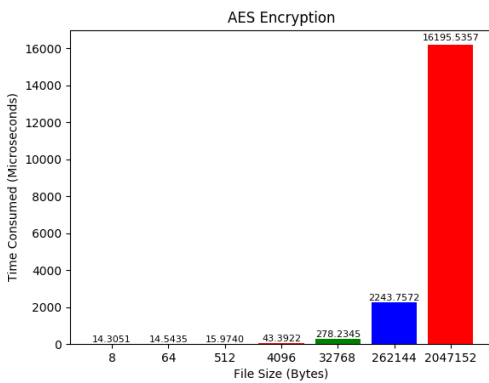
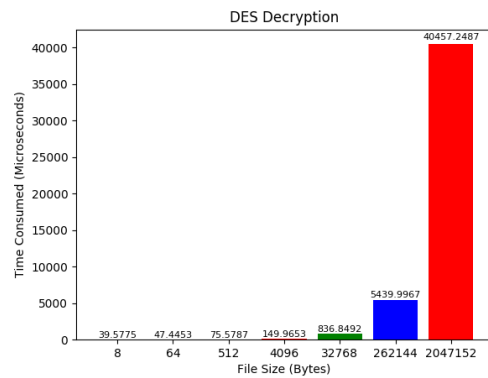
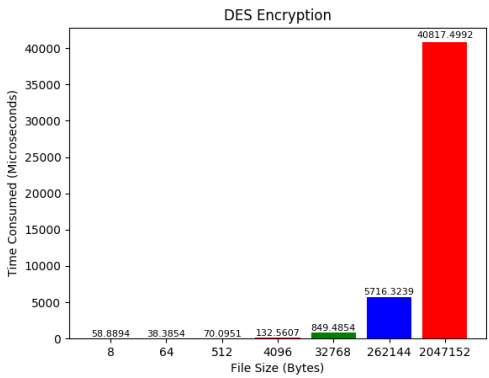
In part 4.c and 4.d, all functions are included in run.py, following such steps:

1. It uses python built-in random method to generate random characters in original test files in different sizes.
2. Then it calls each encryption method (in different python files) to encrypt and/or decrypt such test files.
3. As each method returns encryption and/or decryption time, it records the time consumption for each method in using test files of different sizes.
4. Use matplotlib module to visualize the relation of time consumption and file size, in different methods.

We use such method to read test files in different sizes.

1. Input file as binary format
2. Calculating the file length, if the length of file is not multiple of 8 ,then we append extra byte in the end of the file by using '\x00' string.
3. If the length of file is multiple of 8 , then we implement the algorithm to encrypt it directly.

GRAPHS



QUESTIONS

1. Compare DES encryption and AES encryption. Explain your observations.

From the graphs, it is clear that DES encryption takes more time than AES encryption. Typically, consider within file size 2047152 Bytes, DES (40817 μ s) is about three-times inefficient than AES (16195 μ s).

Thus, DES is comparatively slower than AES. This is because DES uses the Feistel network with permutation encryption steps, while AES uses permutation-substitution steps to encrypt.

2. Compare DES encryption and RSA encryption. Explain your observations.

As the sizes of test files between DES and RSA are not all the same, we can just compare the time consumption within the file (size 8-Bytes and 64-Bytes) only.

DES takes 58 μ s to encrypt 8-Bytes file, while RSA takes 90 μ s.

DES takes 38 μ s to encrypt 64-Bytes file, while RSA takes 128 μ s.

Thus, DES is timely more efficient than RSA, that's mainly because DES is a symmetric method, but RSA is an asymmetric method.

3. Compare DES encryption and SHA-1 digest generation. Explain your observations.

DES runs much slower than SHA-1 digest. for the file size 2047152 Bytes, DES (40817 μ s) is considerably inefficient than SHA-1 (12 μ s).

This is because SHA-1 is a hash-digest algorithm, which is generally faster than symmetric encryption method.

4. Compare HMAC signature generations and SHA-1 digest generation. Explain your observations.

In my test result, SHA-1 is slightly efficient than HMAC, with the time difference up to 7 μ s. The maximum difference is shown in 8-Bytes test file, SHA-1 runs 6 μ s, while HMAC runs 13 μ s.

I think this is possibly because HMAC uses salt strings, once the length of input string is less than a block length, padding is required, which results in more time consumption.

5. Compare RSA encryption and decryption times. Can you explain your observations?

According to the figures, RSA encryption is much faster than its decryption. (ten-times faster in average).

The reason is, RSA is based on large prime number decomposition. When encrypting, multiple large prime numbers can be less time-consuming, but decryption requires decomposition, which takes more time.