COMP4337 - Securing Wireless Networks

Lab 1

Intro to Basic Cryptographic Mechanisms

Chris Joy (z5113243), Sen Li (z5183990)
9th of March, 2020

Table of Contents

Source Code

SFWN_AQ.des.py

```
from Crypto.Cipher import DES
import sys
import time

iv = bytes.fromhex(sys.argv[1])
key = bytes.fromhex(sys.argv[2])
inputfiledir = sys.argv[3]
outputfiledir = sys.argv[4]

print('='*100)
print('Key used: ', [x for x in key])
print("IV used: ",[x for x in iv])
print('='*100)

des1 = DES.new(key, DES.MODE_CBC, iv)
des2 = DES.new(key, DES.MODE_CBC, iv)

with open(inputfiledir, 'rb') as infile:
   plain_text = infile.read()
   # pad to make last block length a multiple of 8
   padded_text = plain_text
   remainder = len(plain_text) % 8
   if remainder > 0:
       length = 8 - remainder
       padded_text += b'\x00'*length
   # encrypt padded text
   start_time = time.time()
   cipher_text = des1.encrypt(padded_text)
   encryption_time = time.time() - start_time

with open(outputfiledir, 'wb') as outfile:
   outfile.write(cipher_text)
   outfile.close()

print('Plaintext is:', plain_text)
print('Ciphertext is:', cipher_text)
start_time = time.time()
original_msg = des2.decrypt(cipher_text)
decryption_time = time.time() - start_time
print('Original Message:', original_msg.decode('utf-8'))
print('='*100)
print('Encryption time:', encryption_time * 1000000)
```

```
print('Decryption time:', decryption_time * 1000000)
```

SFWN_AQ.aes.py

```
from Crypto.Cipher import AES
import sys
import time

iv = bytes.fromhex(sys.argv[1])
key = bytes.fromhex(sys.argv[2])
inputfiledir = sys.argv[3]
outputfiledir = sys.argv[4]

print('='*100)
print('Key used: ', [x for x in key])
print("IV used: ",[x for x in iv])
print('='*100)

aes1 = AES.new(key, AES.MODE_CBC, iv)
aes2 = AES.new(key, AES.MODE_CBC, iv)

with open(inputfiledir, 'rb') as infile:
   plain_text = infile.read()
   # pad to make last block length a multiple of 16
   padded_text = plain_text
   remainder = len(plain_text) % 16
   if remainder > 0:
       length = 16 - remainder
       padded_text += b'\x00'*length
   # encrypt padded text
   start_time = time.time()
   cipher_text = aes1.encrypt(padded_text)
   encryption_time = time.time() - start_time

with open(outputfiledir, 'wb') as outfile:
   outfile.write(cipher_text)
   outfile.close()

print('Plaintext is:', plain_text)
print('Ciphertext is:', cipher_text)
start_time = time.time()
original_msg = aes2.decrypt(cipher_text)
decryption_time = time.time() - start_time
print('Original Message:', original_msg.decode('utf-8'))
```

```
print('='*100)
print('Encryption time:', encryption_time * 1000000)
print('Decryption time:', decryption_time * 1000000)
```

SFWN_AQ.RSA.py

```
import Crypto
from Crypto.PublicKey import RSA
from Crypto import Random
import ast
import sys
import time
inputfiledir = sys.argv[1]
timefiledir = sys.argv[2]
random_generator = Random.new().read
key = RSA.generate(2048, random_generator)
publickey = key.publickey()
print('='*100)
input = open(inputfiledir,"r")
plain_text = input.read()
encrypt_start = time.time()
cipher_text = publickey.encrypt(plain_text, 32)
encrypt_end = time.time()
print ('Plaintext encrypted using Public Key is:', cipher_text)
print
#decrypted code below
decrypt_start = time.time()
decrypted = key.decrypt(ast.literal_eval(str(cipher_text)))
decrypt_end = time.time()
print ('Ciphertext decrypted with Private key is', decrypted)
print ('='*100)
with open(timefiledir, 'a') as timefile:
   timefile.write("encrypt time = ")
   timefile.write(str((encrypt_end - encrypt_start)*1000000))
   timefile.write("\n")
   timefile.write("decrypt time = ")
   timefile.write(str((decrypt_end - decrypt_start)*1000000))
   timefile.write("\n")
```

files/gen_files.sh (used to generate data for graphs)

```
array=( 8 64 512 4096 32768 262144 2047152 )
for i in "${array[@]}"
```

```
do
   echo Creating ADES 1 file of size: $i bytes
   dd if=/dev/zero of=ades_$i.txt count=1 bs=$i
done



array=( 2 4 8 16 32 64 128 )
for i in "${array[@]}"
do
   echo Creating RSA 1 file of size: $i bytes
   dd if=/dev/zero of=rsa_$i.txt count=1 bs=$i
done
```

test_algos.sh (used to collect timing data for graphs)

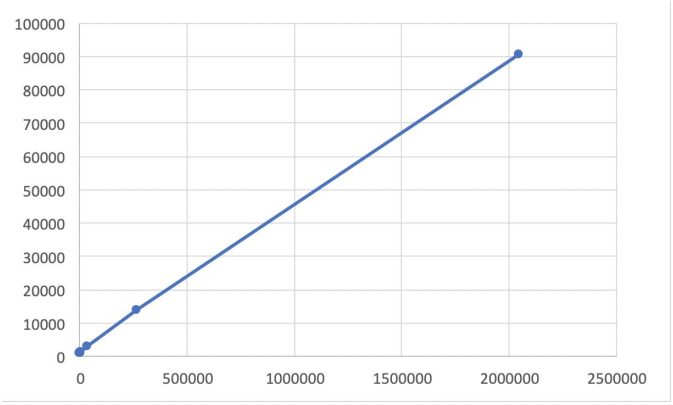
```
echo ------------------------------------------------
echo DES
echo ------------------------------------------------
array=( 8 64 512 4096 32768 262144 2047152 )
for i in "${array[@]}"
do
   echo bytes: $i
   python3 SFWN_AQ.des.py fedcba9876543210 40fedf386da13d57 files/ades_$i.txt mytest.des |
tail -2
   echo ------------------------------------------------
done



echo ------------------------------------------------
echo AES
echo ------------------------------------------------
array=( 8 64 512 4096 32768 262144 2047152 )
for i in "${array[@]}"
do
   echo bytes: $i
   python3 SFWN_AQ.aes.py fedcba9876543210fedcba9876543210
40fedf386da13d5740fedf386da13d57 files/ades_$i.txt mytest.aes | tail -2
   echo ------------------------------------------------
done
```

Graphs

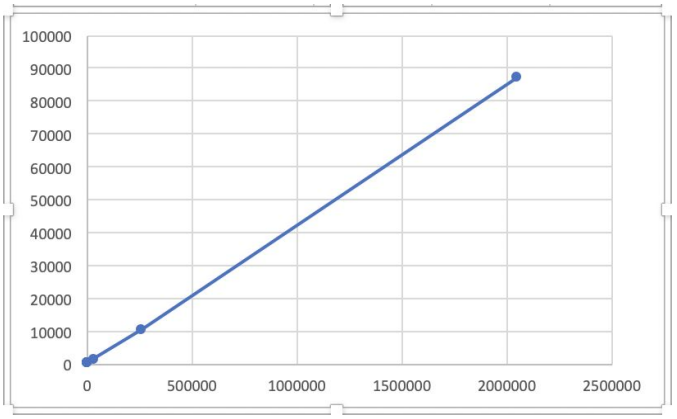
In each of these graphs, the X-axis should plot the file sizes in units of bytes, and the Y-axis should plot time measurements in units of microseconds (μs).

1) DES encryption / decryption times

Encryption

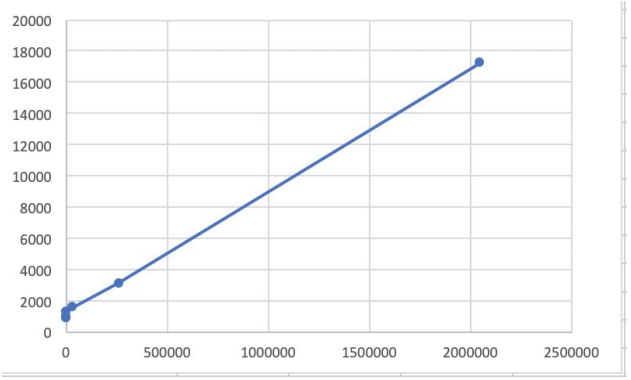


Decryption

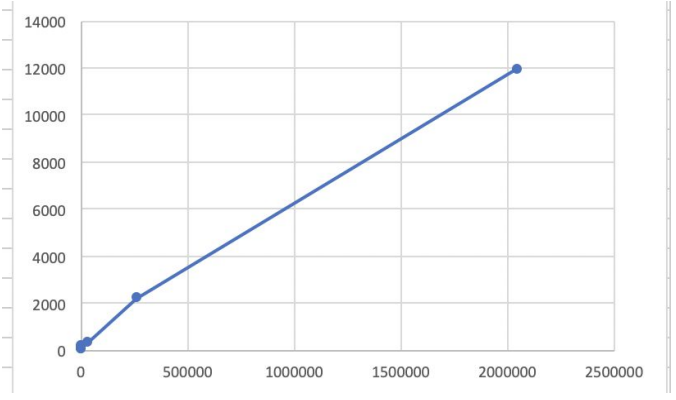


2) AES encryption / decryption times

Encryption

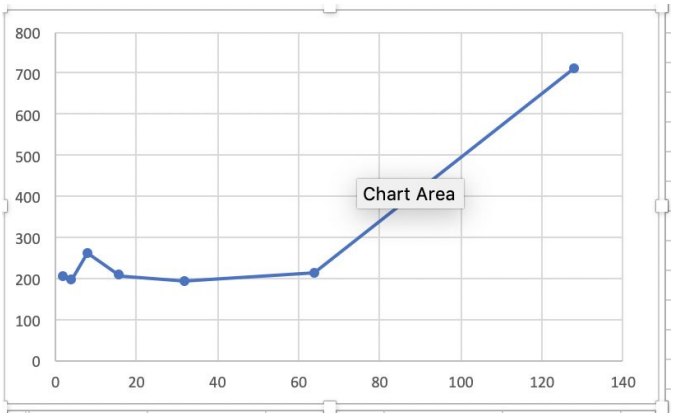


Decryption

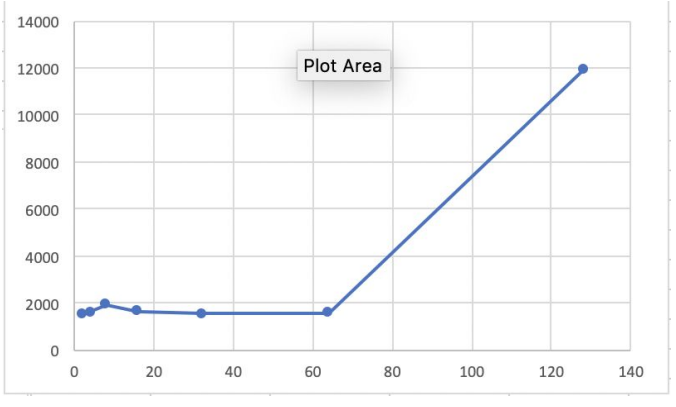


3) RSA encryption / decryption times

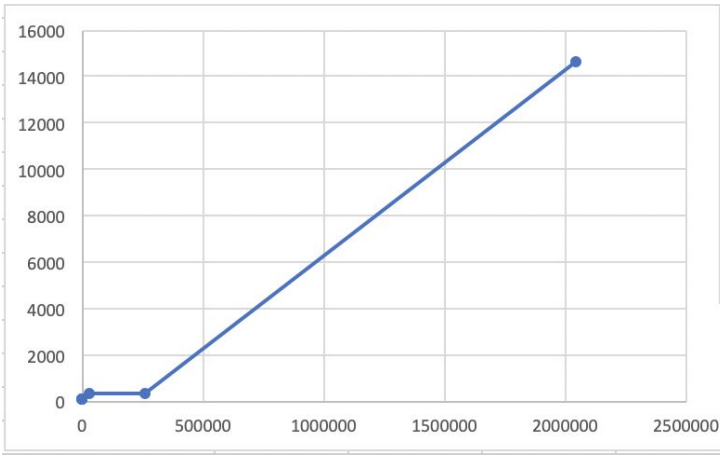
Encryption



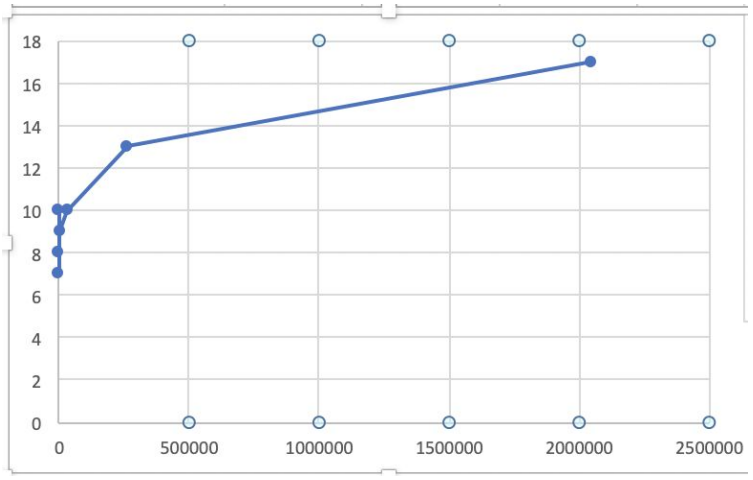
Decryption



4) SHA-1 digest generation times



5) HMAC signature generation times.



Questions

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

- DES on average takes longer than AES when encrypting and decrypting.
- On average, when encrypting 2047152 bytes:
    - DES takes 27303 µs
    - AES takes 5132 µs
    - We can see that AES is approximately 5 times faster than DES, when encrypting.
- On average, when decrypting 2047152 bytes:
    - DES takes 23438 µs
    - AES takes 3288 µs
    - We can see that AES is approximately 7 times faster than DES, when decrypting.

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

- RSA is slower on average compared with DES, when encrypting and decrypting.
- On average, when encrypting 8 bytes:
    - DES takes 847 µs
    - RSA takes 1937 µs
    - We can see that DES is approximately 2 times faster than RSA, when encrypting.

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

- SHA-1 digest generation is much faster than DES encryption.
- On average, when deal with 4096 bytes file:
    - DES takes 861 µs
    - SHA-1 takes 56 µs
    - SHA-1 is approximately 15 times faster.
- DES is a symmetric encryption method whereas SHA-1 is a hashing algorithm providing the digest, given a message. They perform non-equivalent operations and should not be compared.

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

- SHA-1 digest generation is slower than HMAC signature generation.
- On average when deal with 2047152 bytes file:
    - HMAC takes 17 µs
    - SHA-1 takes 14625 µs

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

- RSA encryption is significantly faster compared to when decrypting.
- In RSA encryption the public key is chosen manually, whereas the secret key is derived from using modulo arithmetic on the public key. Normally we use a smaller public key, making the encryption process faster.
- On average when decrypting and encrypting 4096 bytes:
    - Encryption takes 260 µs
    - Decryption takes 1934 µs