COMP9337 21T1
LAB 1 – BASIC CRYPTOGRAPHY

Gordon Chen (z5161163)
Eu Shaun Lim (z5156345)

Code

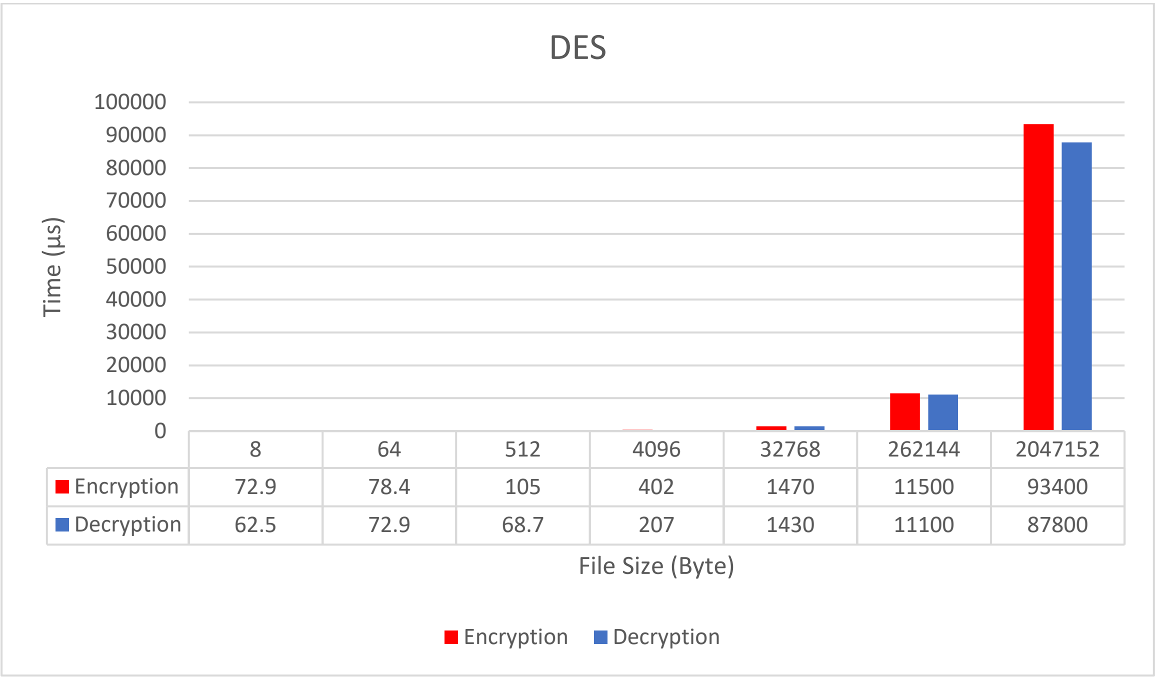
```
from Crypto.Cipher import DES
from Crypto import Random
from binascii import unhexlify
import sys
import time

def is_hex(s):
  try:
    int(s, 16)
    return True
  except ValueError:
    return False

# Get and check iv and cbc
iv = sys.argv[1]
cbc_key = sys.argv[2]

if not is_hex(iv):
  print("Error: IV is not in hexadecimal string")
  sys.exit()

if not is_hex(cbc_key):
  print("Error: Key is not in hexadecimal string")
  sys.exit()

# Convert hex to binary
cbc_key = unhexlify(sys.argv[2])
print('='*100)
print('Key used: ', [x for x in cbc_key])

iv = unhexlify(sys.argv[1])
print("IV used: ", [x for x in iv])
print('='*100)

# Open file
try:
  file_name = sys.argv[3]
  f = open(file_name,'rb')
  plain_text = f.read()
except:
  sys.exit()

# Padding
while len(plain_text) != 0:
        plain_text = plain_text.__add__(b'\x00')
```

```
# Encrypt
start_time1 = time.time()
des1 = DES.new(cbc_key, DES.MODE_CBC, iv)
cipher_text = des1.encrypt(plain_text)
encrypt_time = time.time() - start_time1
#print("Ciphertext is: ",cipher_text)

# Write to output file
with open(sys.argv[4], 'wb') as f:
    f.write(cipher_text)

# Decrypt
start_time2 = time.time()
des2 = DES.new(cbc_key, DES.MODE_CBC, iv)
msg = des2.decrypt(cipher_text)
decrypt_time = time.time() - start_time2
#print("Original Message", msg)
#print('='*100)

print("Encryption time is: ", encrypt_time)
print("Decryption time is: ", decrypt_time)
```

Graph

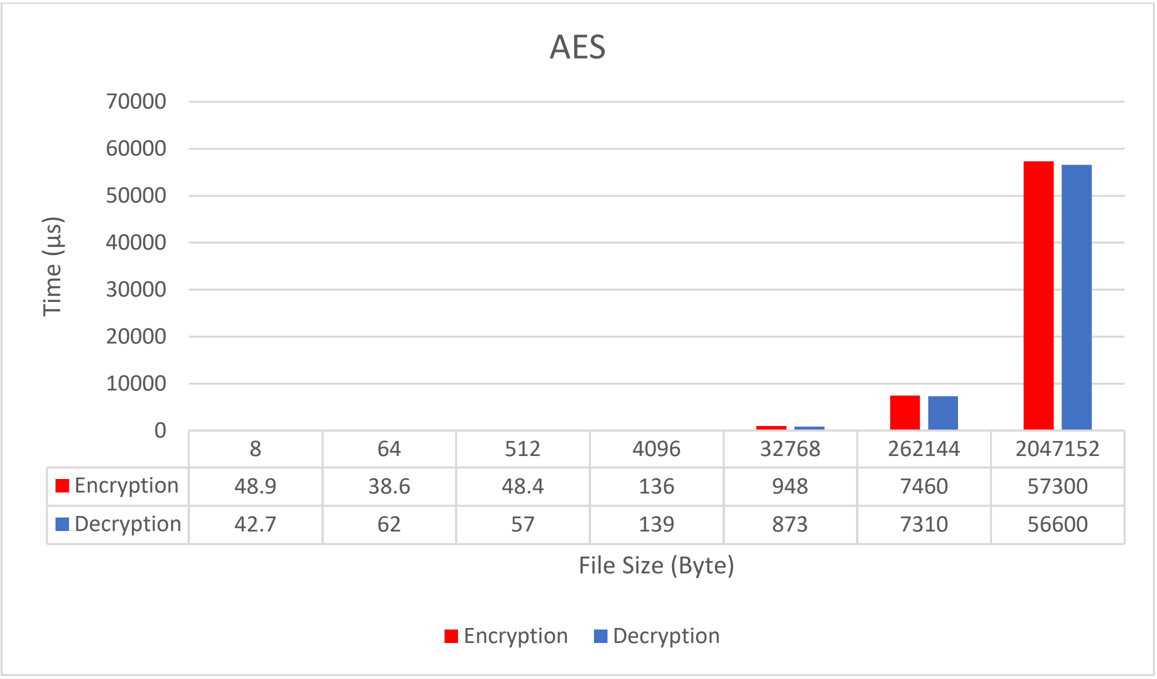
In graphs, red means encryption and blue means decryption

DES Encryption/Decryption Time



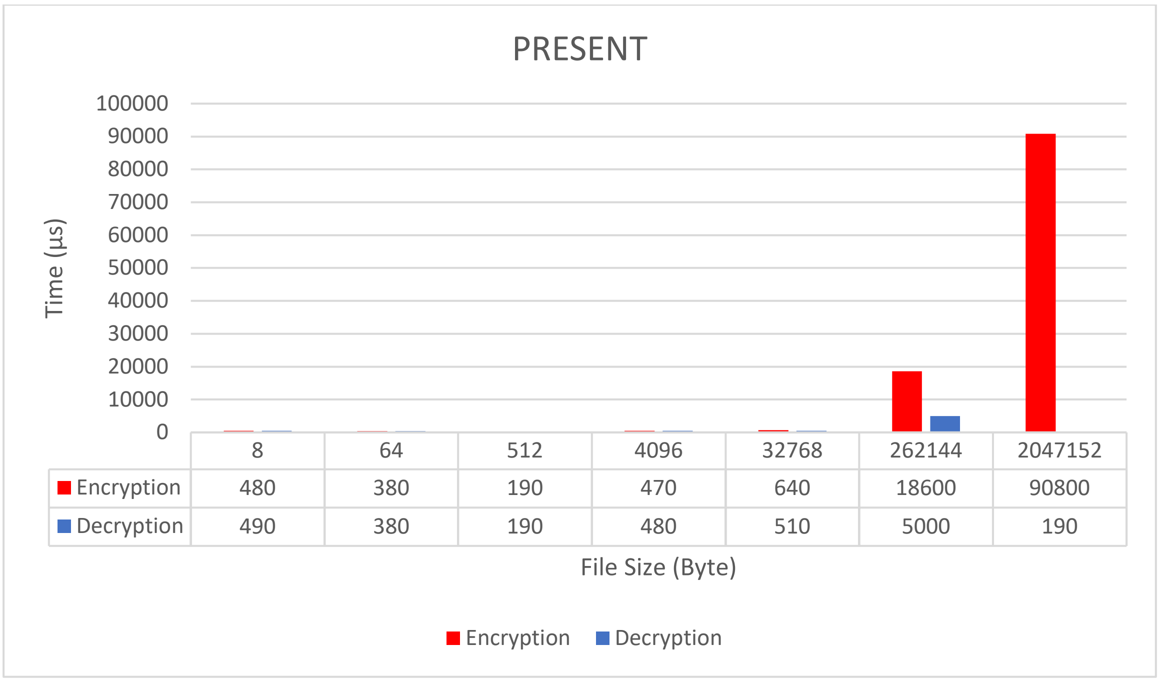
	8	64	512	4096	32768	262144	2047152
🟥 Encryption	72.9	78.4	105	402	1470	11500	93400
🟦 Decryption	62.5	72.9	68.7	207	1430	11100	87800

AES



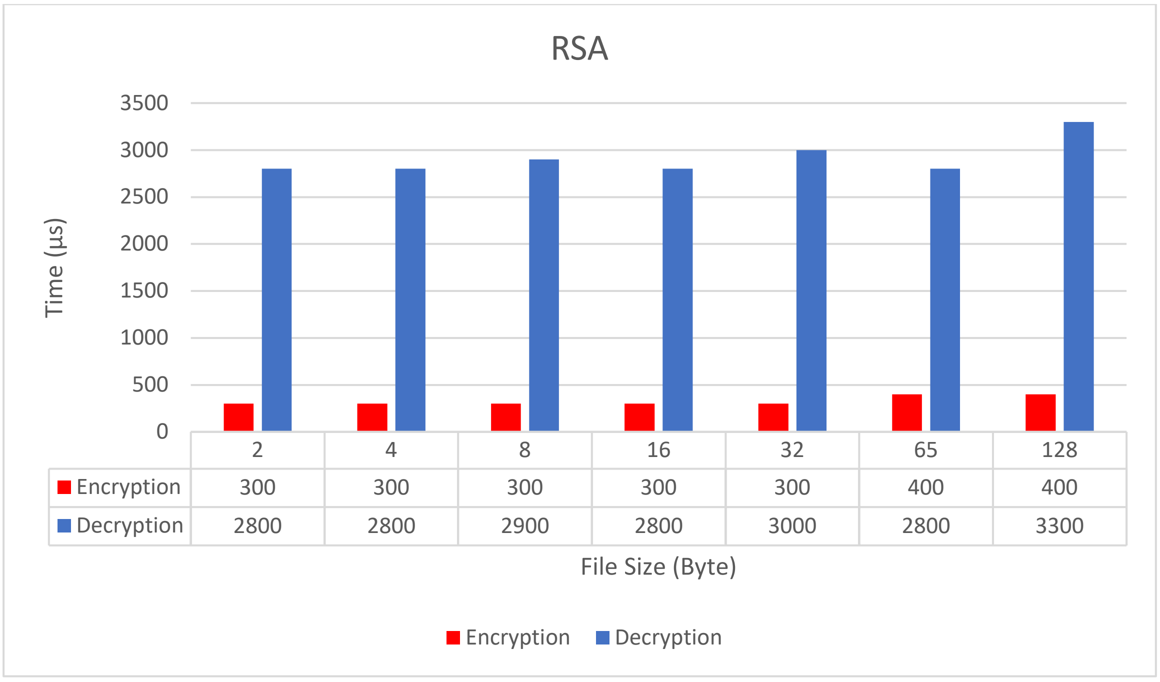
	8	64	512	4096	32768	262144	2047152
🟥 Encryption	48.9	38.6	48.4	136	948	7460	57300
🟦 Decryption	42.7	62	57	139	873	7310	56600

PRESENT



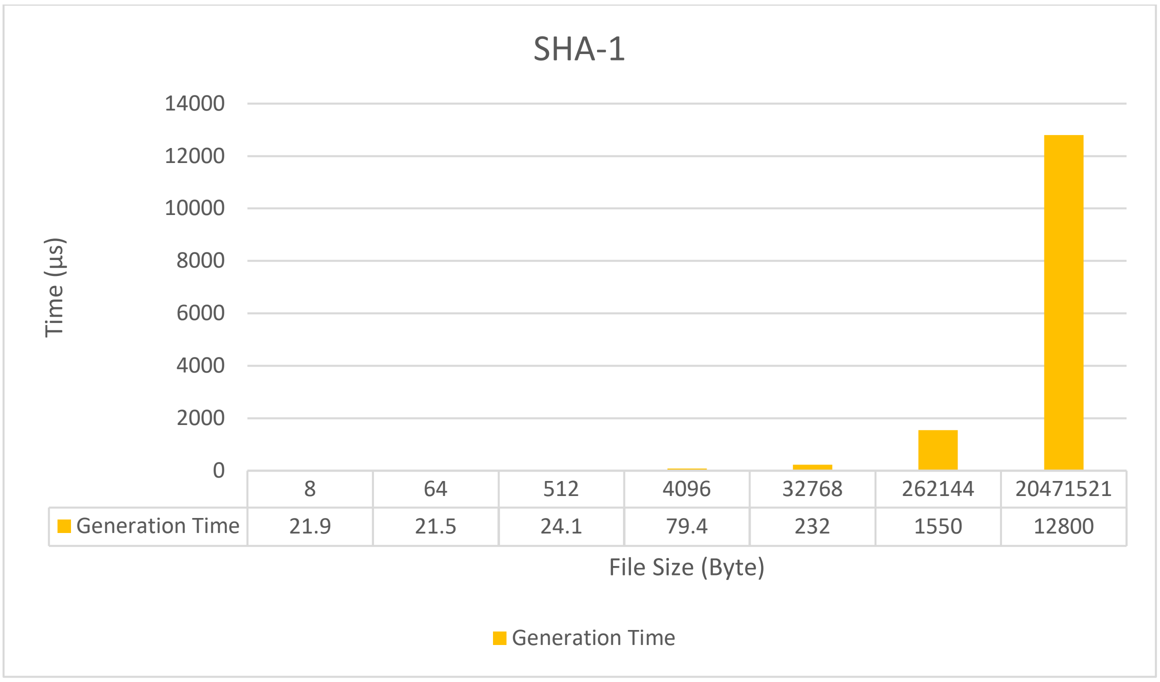
	8	64	512	4096	32768	262144	2047152
🟥 Encryption	480	380	190	470	640	18600	90800
🟦 Decryption	490	380	190	480	510	5000	190

RSA



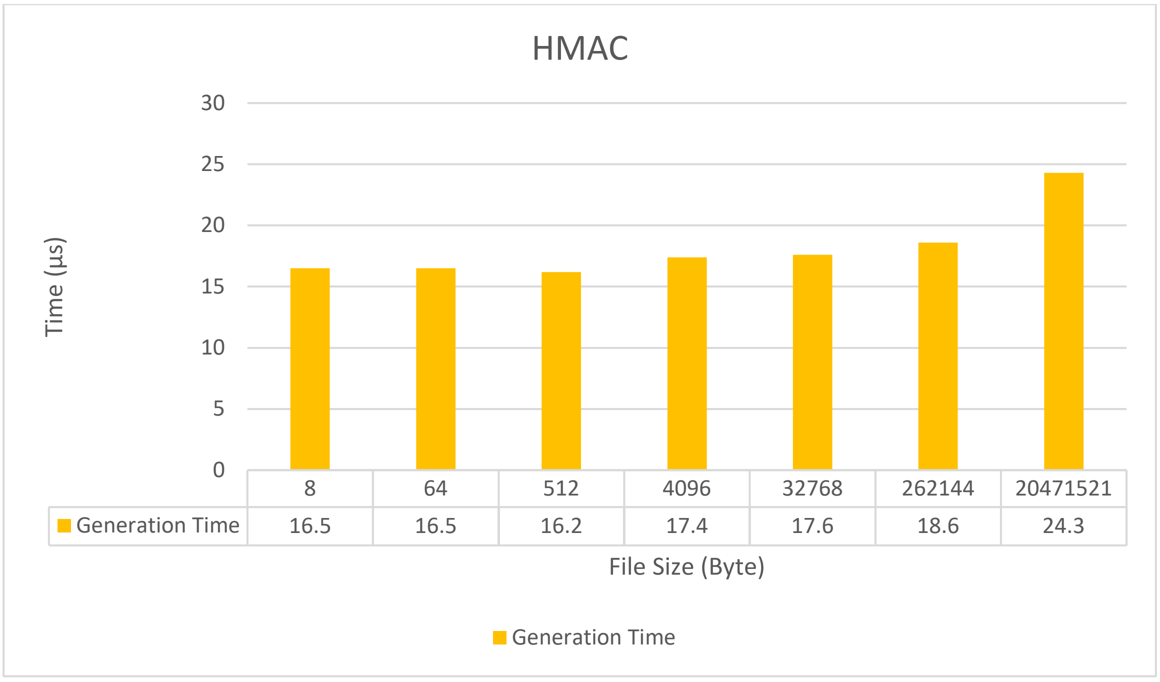
	2	4	8	16	32	65	128
🟥 Encryption	300	300	300	300	300	400	400
🟦 Decryption	2800	2800	2900	2800	3000	2800	3300

SHA-1



	8	64	512	4096	32768	262144	20471521
🟨 Generation Time	21.9	21.5	24.1	79.4	232	1550	12800

HMAC



	8	64	512	4096	32768	262144	20471521
🟨 Generation Time	16.5	16.5	16.2	17.4	17.6	18.6	24.3

Observation

Compare DES and AES

From the figures, AES is more efficient that DES which is more evident in large file size. Although AES uses more key and block size than DES, AES data encryption is shown to be more mathematically efficient and more secure algorithm compared to DES. And those facts are shown in the last 2 examples where there is a noticeable gap in both encryption and decryption time between DES and AES.

Compare DES encryption and PRESENT encryption

From the figures, there are different growth rate between both algorithms in encryption. In very small files, DES is shown to be more efficient than PRESENT. However, in medium and large file size, there are similar performance result between those 2 algorithms. In fact, PRESENT algorithm is more efficient at encrypting files of size 32768 and 2047152 bytes.

Compare DES and RSA

From the figures, DES is more efficient than RSA in encryption efficiency for small files. A possible reason could be the long key of RSA algorithm which means the encryption file is very large and slow. However, for very large file, RSA time to encrypt remains consistent at 400 milliseconds while DES struggles mightily.

For decryption, it is the same case. In small files, DES is more efficient than RSA due to the long key from RSA algorithm. However, RSA time remains consistent throughout the whole files test while DES exponentially increases its time as the file size increases.

Compare DES and SHA-1

From the figures, SHA-1 is shown to be faster than DES in all file sizes. However, it should be noted that both algorithms are different as SHA-1 algorithm is used to apply data integrity while DES is used to encrypted plain text. As a result, it makes sense that SHA-1 algorithm has less consuming tasks and hence has a faster performance.

Compare HMAC and SHA-1

From the figures, HMAC has a higher efficiency than SHA-1. It should be noted that HMAC is based on MD5 which is shown to be more efficient than SHA-1.

Compare RSA encryption and decryption time

From the figures, RSA encryption is faster than decryption for all file sizes. This is because public key is a lot shorter in length than private key in RSA algorithm. Hence, it is quick to encrypt the text with a smaller public key while it’s slower for decrypting as it involves big numbers due to larger private key, heavily reducing the efficiency of decryption.