密碼學作業一

409410050 王謙靜

程式檔案架構

1. `plaintext_gen.py` 是用來產生 `plain_text.txt` 的
2. 三種加密法個放在一個資料夾中，分別有加密與解密的程式
3. `run.sh` 可以用來執行三個密碼的加解密，正確性檢查以及時間的測量

AESCBC

encrypt

```
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad

f = open("../plaintext.txt", "r")
message = f.read().encode()
random_key = get_random_bytes(16)
f = open("key.txt", "wb")
f.write(random_key)

key = random_key
cipher = AES.new(key, AES.MODE_CBC)
cipher_byte = cipher.encrypt(pad(message, AES.block_size))
initial_vector = cipher.iv;

f = open("cipher.txt", "wb")
f.write(cipher_byte)
f = open("init_vector.txt", "wb")
f.write(initial_vector)
```

decrypt

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

f = open("key.txt", "rb")
key = f.read()
f = open("init_vector.txt", "rb")
init_vector = f.read()
f = open("cipher.txt", "rb")
ct = f.read()

cipher = AES.new(key, AES.MODE_CBC, init_vector)
pt = unpad(cipher.decrypt(ct), AES.block_size)

f = open("retrive.txt", "w")
f.write(pt.decode())
```

AESCTR

encrypt

```
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad

f = open("../plaintext.txt", "r")
message = f.read().encode()
random_key = get_random_bytes(16)
f = open("key.txt", "wb")
f.write(random_key)

key = random_key
cipher = AES.new(key, AES.MODE_CTR)
cipher_byte = cipher.encrypt(message)
nonce = cipher.nonce;

f = open("cipher.txt", "wb")
f.write(cipher_byte)
f = open("nonce.txt", "wb")
f.write(nonce)
```

decrypt

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

f = open("key.txt", "rb")
key = f.read()
f = open("nonce.txt", "rb")
nonce = f.read()
f = open("cipher.txt", "rb")
ct = f.read()

cipher = AES.new(key, AES.MODE_CTR, nonce=nonce)
pt = cipher.decrypt(ct)

f = open("retrive.txt", "w")
f.write(pt.decode())
```

ChaCha20

encrypt

```
from Crypto.Random import get_random_bytes
from Crypto.Cipher import ChaCha20
from Crypto.Util.Padding import pad

f = open("../plaintext.txt", "r")
message = f.read().encode()
random_key = get_random_bytes(32)
f = open("key.txt", "wb")
f.write(random_key)

key = random_key
cipher = ChaCha20.new(key=key)
cipher_byte = cipher.encrypt(message)
nonce = cipher.nonce;

f = open("cipher.txt", "wb")
f.write(cipher_byte)
f = open("nonce.txt", "wb")
f.write(nonce)
```

decrypt

```
from Crypto.Cipher import ChaCha20
from Crypto.Util.Padding import unpad

f = open("key.txt", "rb")
key = f.read()
f = open("nonce.txt", "rb")
nonce = f.read()
f = open("cipher.txt", "rb")
ct = f.read()

cipher = ChaCha20.new(key=key, nonce=nonce)
pt = cipher.decrypt(ct)

f = open("retrive.txt", "w")
f.write(pt.decode())
```

被加密的檔案大小

被加密的檔案大小 191MB

```
forward@forward-System-Product-Name:~/class/Crypto/Network/HW1$ ls -alh
total 191M
drwxrwxr-x 5 forward forward 4.0K Mar 23 15:12 .
drwxrwxr-x 4 forward forward 4.0K Mar 23 13:16 ..
drwxrwxr-x 2 forward forward 4.0K Mar 23 15:19 AESCBC
drwxrwxr-x 2 forward forward 4.0K Mar 23 15:14 AESCTR
drwxrwxr-x 2 forward forward 4.0K Mar 23 15:14 ChaCha20
-rw-rw-r-- 1 forward forward  301 Mar 23 15:12 plaintext_gen.py
-rw-rw-r-- 1 forward forward 191M Mar 23 15:13 plaintext.txt
-rwxrwxr-x 1 forward forward  264 Mar 23 15:06 run.sh
```

執行畫面

我寫了一個 `run.sh` 的 script 來執行加解密、時間測量以及檢查與一開始加密的內容是否一樣。

run.sh

```
cipher=("AESCBC" "AESCTR" "ChaCha20")

for i in ${cipher[@]}
do
    cd $i
    echo "running $i"
    rm *.txt
    time python3 encrypt.py
    python3 decrypt.py
    echo "diff retrive.txt ../plaintext.txt"
    diff retrive.txt ../plaintext.txt
    cd ../
done
```

執行結果如下

```
forward@forward-System-Product-Name:~/class/Crypto/Network/HW1$ ./run.sh
running AESCBC

real    0m1.008s
user    0m0.479s
sys     0m0.529s
diff retrive.txt ../plaintext.txt
running AESCTR

real    0m0.846s
user    0m0.402s
sys     0m0.445s
diff retrive.txt ../plaintext.txt
running ChaCha20

real    0m1.048s
user    0m0.593s
sys     0m0.456s
diff retrive.txt ../plaintext.txt
forward@forward-System-Product-Name:~/class/Crypto/Network/HW1$
```

三種加密方式的速度

精確來說 plaintext.txt 有 $200,000,000$ 個 bytes

故 AESCBC 平均一秒可以加密 $\frac{2 \cdot 10^8}{1.008} = 1.984 \times 10^8$ 個 bytes

故 AESCTR 平均一秒可以加密 $\frac{2 \cdot 10^8}{0.846} = 2.364 \times 10^8$ 個 bytes

故 ChaCha20 平均一秒可以加密 $\frac{2 \cdot 10^8}{1.048} = 1.908 \times 10^8$ 個 bytes

比較解密後的檔案與原始檔案

```
forward@forward-System-Product-Name:~/class/Crypto/Network/HW1$ diff plaintext.txt AESCBC/retrive.txt
forward@forward-System-Product-Name:~/class/Crypto/Network/HW1$ diff plaintext.txt AESCTR/retrive.txt
forward@forward-System-Product-Name:~/class/Crypto/Network/HW1$ diff plaintext.txt ChaCha20/retrive.txt
forward@forward-System-Product-Name:~/class/Crypto/Network/HW1$
```

reference

1. https://pycryptodome.readthedocs.io/en/latest/src/cipher/classic.html#cbc-mode (https://pycryptodome.readthedocs.io/en/latest/src/cipher/classic.html#cbc-mode)
2. https://pycryptodome.readthedocs.io/en/latest/src/cipher/chacha20.html?highlight=ChaCha20 (https://pycryptodome.readthedocs.io/en/latest/src/cipher/chacha20.html?highlight=ChaCha20)
3. https://www.w3schools.com/python/python_file_write.asp (https://www.w3schools.com/python/python_file_write.asp)
4. https://www.geeksforgeeks.org/python-write-bytes-to-file/ (https://www.geeksforgeeks.org/python-write-bytes-to-file/)