密碼學 assignment 1

409410014 資工四 柯柏旭

1. 請使用加密相關的library，實作加密檔案的程式(檔案需至少100MB)。

how to run

- 生成100MB的隨機plaintext

```
python3 gen_data.py
```

- AES-CCM 加密 (測速和比對功能都在裡面)

```
python3 AES-CCM.py
```

- AES-CTR 和 ChaCha20 加密.py (測速和比對功能都在裡面)

```
python3 main.py
```

- 問題2

```
python3 prob2.py
```

(1) 原始程式碼與說明被加密的檔案大小

- 生成檔案的code:

```
def generate_large_file(filename, size):
    chars = string.ascii_letters + string.digits
    with open(filename, 'w') as file:
        while os.path.getsize(filename) < size:
            file.write(''.join(random.choice(chars) for _ in range(1024)))

filename = 'data.txt'
size = 100 * 1024 * 1024  # 100MB

generate_large_file(filename, size)
```

- 檔案大小

```
-rw-rw-r-- 1 hentci hentci 100M Apr 20 17:08  data.txt
```

- AES-CCM mode加密

參數設定:

```
key_size = 16
nonce_size = 10
key = os.urandom(key_size)
nonce = os.urandom(nonce_size)
```

```
def encrypt_decrypt_aes_ccm(key, nonce, filename, encrypted_filename,
decrypted_filename):
    # 初始化 AESCCM 對象
    cipher = AESCCM(key)

    with open(filename, 'rb') as f:
        plaintext = f.read()

    # 加密
    start_time = time.time()
    ciphertext = cipher.encrypt(nonce, plaintext, None)
    end_time = time.time()
    encryption_speed = len(plaintext) / (end_time - start_time)

    with open(encrypted_filename, 'wb') as f:
        f.write(ciphertext)

    # 解密
    start_time = time.time()
    decrypted_data = cipher.decrypt(nonce, ciphertext, None)
    end_time = time.time()
    decryption_speed = len(ciphertext) / (end_time - start_time)

    with open(decrypted_filename, 'wb') as f:
        f.write(decrypted_data)

    assert plaintext == decrypted_data, "Decrypted data does not match original!"

    if plaintext == decrypted_data:
        print("AES-CCM - Encryption and decryption successful! Data integrity
maintained.")

    return encryption_speed, decryption_speed
```

被加密的檔案大小:

```
-rw-rw-r-- 1 hentci hentci 101M Apr 20 18:58  data.txt.aes.ccm.encrypted
```

- AES-CTR mode加密 (p.s. AES-CTR 和 chacha20 寫在 function，所以 code 內容一樣)

參數設定:

```
key_size = 16
nonce_size = 16
key = os.urandom(key_size)
nonce = os.urandom(nonce_size)
```

```
def encrypt_decrypt(algorithm, mode, key, nonce, filename, encrypted_filename,
decrypted_filename):
    if algorithm == 'AES' and mode == 'CTR':
        cipher = Cipher(algorithms.AES(key), modes.CTR(nonce),
backend=default_backend())
        encryptor = cipher.encryptor().update
        decryptor = cipher.decryptor().update
    elif algorithm == 'ChaCha20':
        cipher = ChaCha20Poly1305(key)
        encryptor = lambda data: cipher.encrypt(nonce, data, None)
        decryptor = lambda data: cipher.decrypt(nonce, data, None)

    with open(filename, 'rb') as f:
        plaintext = f.read()


    start_time = time.time()
    ciphertext = encryptor(plaintext)
    end_time = time.time()
    encryption_speed = len(plaintext) / (end_time - start_time)

    with open(encrypted_filename, 'wb') as f:
        f.write(ciphertext)

    start_time = time.time()
    decrypted_data = decryptor(ciphertext)
    end_time = time.time()
    decryption_speed = len(ciphertext) / (end_time - start_time)

    with open(decrypted_filename, 'wb') as f:
        f.write(decrypted_data)

    assert plaintext == decrypted_data, "Decrypted data does not match original!"

    if plaintext == decrypted_data:
        print(f"{algorithm} - Encryption and decryption successful! Data
integrity maintained.")

    return encryption_speed, decryption_speed
```

被加密的檔案大小:

```
-rw-rw-r-- 1 hentci hentci 100M Apr 20 18:48  data.txt.aes.ctr.encrypted
```

- ChaCha20 加密 (p.s. AES-CTR 和 chacha20 寫在 function，所以 code 內容一樣)

參數設定:

```
key_size = 32
nonce_size = 12
key = os.urandom(key_size)
nonce = os.urandom(nonce_size)
```

```
def encrypt_decrypt(algorithm, mode, key, nonce, filename, encrypted_filename,
decrypted_filename):
```

```
    if algorithm == 'AES' and mode == 'CTR':
        cipher = Cipher(algorithms.AES(key), modes.CTR(nonce),
backend=default_backend())
        encryptor = cipher.encryptor().update
        decryptor = cipher.decryptor().update
    elif algorithm == 'ChaCha20':
        cipher = ChaCha20Poly1305(key)
        encryptor = lambda data: cipher.encrypt(nonce, data, None)
        decryptor = lambda data: cipher.decrypt(nonce, data, None)

    with open(filename, 'rb') as f:
        plaintext = f.read()


    start_time = time.time()
    ciphertext = encryptor(plaintext)
    end_time = time.time()
    encryption_speed = len(plaintext) / (end_time - start_time)

    with open(encrypted_filename, 'wb') as f:
        f.write(ciphertext)

    start_time = time.time()
    decrypted_data = decryptor(ciphertext)
    end_time = time.time()
    decryption_speed = len(ciphertext) / (end_time - start_time)

    with open(decrypted_filename, 'wb') as f:
        f.write(decrypted_data)

    assert plaintext == decrypted_data, "Decrypted data does not match original!"

    if plaintext == decrypted_data:
        print(f"{algorithm} - Encryption and decryption successful! Data
integrity maintained.")

    return encryption_speed, decryption_speed
```

被加密的檔案大小:

```
-rw-rw-r-- 1 hentci hentci 101M Apr 20 18:48  data.txt.chacha20.encrypted
```

(2) 分別執行以上三種加密方式的速度 (每秒可加密多少 bytes)

```
    with open(filename, 'rb') as f:
        plaintext = f.read()

    # 加密
    start_time = time.time()
    ciphertext = cipher.encrypt(nonce, plaintext, None)
    end_time = time.time()
    encryption_speed = len(plaintext) / (end_time - start_time)
```

```
    with open(encrypted_filename, 'wb') as f:
        f.write(ciphertext)

    # 解密
    start_time = time.time()
    decrypted_data = cipher.decrypt(nonce, ciphertext, None)
    end_time = time.time()
    decryption_speed = len(ciphertext) / (end_time - start_time)
```

```
AES-CCM - Encryption speed: 827353645.94 bytes/sec, Decryption speed: 825741372.28 bytes/sec
('AES', 'CTR') - Encryption speed: 1073042355.47 bytes/sec, Decryption speed: 1060695476.80 bytes/sec
('ChaCha20', None) - Encryption speed: 863635500.01 bytes/sec, Decryption speed: 861046985.52 bytes/sec
```

只比較加密速度的話，分別是:

- AES-CCM: 827353645.94 bytes/sec
- AES-CTR: 1073042355.47 bytes/sec
- ChaCha20: 863635500.01 bytes/sec

即 AES-CTR > ChaCha20 > AES-CCM

(3) 比較解密後的檔案與原始檔案，證明實作正確

```
    assert plaintext == decrypted_data, "Decrypted data does not match original!"

    if plaintext == decrypted_data:
        print("AES-CCM - Encryption and decryption successful! Data integrity
maintained.")
```

```
AES-CCM - Encryption and decryption successful! Data integrity maintained.
AES - Encryption and decryption successful! Data integrity maintained.
ChaCha20 - Encryption and decryption successful! Data integrity maintained.
```

2. 請計算出"I love cryptography." 這個字串（不含雙引號）的SHA-3-512 message digest，並說明你使用的計算工具或程式。

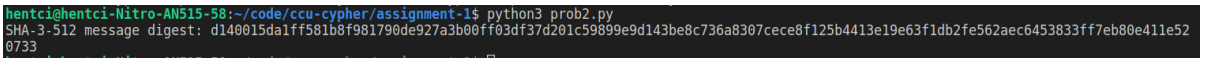
```
import hashlib

# 要計算的字串
message = "I love cryptography."

# 計算 SHA-3-512 message digest
sha3_512_digest = hashlib.sha3_512(message.encode()).hexdigest()

print("SHA-3-512 message digest:", sha3_512_digest)
```

使用`hashlib`的function`hashlib.sha3_512().hexdigest()`就可以得到結果了。



`d140015da1ff581b8f981790de927a3b00ff03df37d201c59899e9d143be8c736a8307cece8f125b4413e19e63f1db2fe562aec6453833ff7eb80e411e520733`