

Assignment #1 Report

컴퓨터소프트웨어학부

2018008395

박정호

1. 실행 환경

A. 사용한 라이브러리

i. python의 hashlib

과제의 두번째 요구사항인 "hash 함수 한가지 이상 사용"을 수행하기 위해, 파이썬의 기본 라이브러리인 hashlib을 사용했다. 사용한 해시 함수는 SHA256, SHA384, SHA512의 세가지이다.

ii. python 의 pycryptodome

과제 명세에서 안내된 pycrypto 는 파이썬 2 기준으로 개발된 라이브러리에 파이썬 3 을 사용하는 필자는 사용할 수 없었다. 게다가 유지보수가 중단된 라이브러리에 파이썬 3 에 맞는 버전을 찾을 수도 없었다.

조금 더 찾아본 결과 pycryptodome 이라는 파이썬 3 기준 라이브러리가 있다는 것을 알게 되었고, 이 라이브러리가 pycrypto 의 개선판이라는 것을 확인해서 이 라이브러리를 사용했다. 사용법은 크게 다르지 않은 걸로 확인되었다.

pip 를 사용해서 다운로드 받았으며, 버전은 아래의 스크린샷에서 확인할 수 있다.

```
pipwin          0.4.7
pycryptodome    3.9.8
pyjssparser     2.7.1
PyOpenGL        3.1.5
```

B. python 버전

사용한 언어는 위에서 언급했듯이 파이썬이며, 버전은 아래의 스크린샷에서 확인할 수 있다.

C. 운영체제

```
C:\Users\pch68>python --version
Python 3.8.2
```

윈도우 10 환경에서 진행했다.

2. 구현

A. 대칭 키 암호화

i. padding 을 통한 text 길이 교정

우선 대칭 키 암호화에서 사용한 block encryption 을 위해서 plaintext 를 align 할 필요가 있었다. 따라서 text_padding 이라는 함수를 구현했다.

```
def text_padding(text, unit):  
    while len(text) % unit != 0:  
        text += '_'.encode('utf-8')  
    return text
```

인자로 사용되는 text 는 기존 plaintext 이고, unit 은 align 할 단위를 의미한다. 따라서 이 함수는 text 의 길이가 unit 의 배수가 될 때까지 '_'를 문자열에 추가하고, 그 결과를 return 한다. 단 여기서 text 는 utf-8 로 인코딩된 bytes object 여야 한다.

ii. DES 방식 암호화

```
def process_DES(text):  
    key = input("key(must be 8 bytes): ").encode('utf-8')  
    while len(key) != 8:  
        print("For DES, key length must be 8 bytes")  
        key = input("key(must be 8 bytes): ").encode('utf-8')  
    des = DES.new(key, DES.MODE_ECB)  
  
    target_text = text_padding(text, 8)  
    encrypted = des.encrypt(target_text)  
    print("encrypted: "+str(encrypted))  
    decrypted = des.decrypt(encrypted)  
    decrypted = decrypted.decode('utf-8')  
    print("decrypted: "+decrypted)
```

라이브러리에서 제공하는 DES 모듈은 key 의 길이를 8 바이트로 강제했다. 따라서 key 의 길이가 8 바이트가 아니라면 다시 입력하라는 메시지와 함께 입력창을 다시 띄우도록 했다. 그렇게 올바른 key 를 입력했다면, 주어진 plaintext 를 8byte-align 되도록 padding 을 추가한다.

모든 준비가 끝났다면, pycryptodome 에서 제공하는 DES 모듈을 사용해서 암호화와 복호화를 진행한다. 모드는 가장 단순한 모드로 배운 ECB 를 사용했다.

iii. DES3 방식 암호화

```
def process_DES3(text):
    des3 = None
    length = None
    while True:
        key = input("key(must be 16 or 24 bytes): ").encode('utf-8')
        length = len(key)
        try:
            key = DES3.adjust_key_parity(key)
            des3 = DES3.new(key, DES3.MODE_ECB)
            break
        except ValueError:
            pass
    target_text = text_padding(text, 8)
    encrypted = des3.encrypt(target_text)
    print("encrypted: "+str(encrypted))
    decrypted = des3.decrypt(encrypted)
    decrypted = decrypted.decode('utf-8')
    print("decrypted: "+decrypted)
```

기본적으로는 DES 와 형식이 거의 같다. 단, pycryptodome 에서 key 의 형식 관련 오류가 총 2 가지가 있어서 try-except 로 오류에 대한 예외처리를 진행했다. 오류는 다음과 같다.

key 의 길이가 16 바이트 혹은 24 바이트가 아닐 경우, 혹은 key 로 인해 Triple DES 가 Single DES 로 degenerate 될 경우에 ValueError 를 throw 하게 된다. 이 Error 는 adjust_key_parity 에서 발생하게 된다. 따라서 ValueError 가 발생할 경우 다시 key 를 입력받도록 바꾸었다.

이후에는 역시 plaintext 를 8byte-align 되도록 padding 을 추가한 뒤, des3 모듈을 사용해서 암호화와 복호화를 진행했다. 여기서도 모드는 ECB 를 사용했다.

iv. AES 방식 암호화

```
def process_AES(text):
    key = input("key(must be 16 or 24 or 32 bytes): ").encode('utf-8')
    while len(key) != 16 and len(key) != 24 and len(key) != 32:
        print("For AES, key length must be 16 or 24 or 32 bytes")
        key = input("key(must be 16 or 24 or 32 bytes): ").encode('utf-8')
    length = len(key)

    aes = AES.new(key, AES.MODE_ECB)
    target_text = text_padding(text, 16)
    encrypted = aes.encrypt(target_text)
    print("encrypted: "+str(encrypted))
    decrypted = aes.decrypt(encrypted)
    decrypted = decrypted.decode('utf-8')
    print("decrypted: "+decrypted)
```

key 의 길이가 16, 24, 32 바이트임을 강제한다는 점을 제외하고는 DES 코드와 완전히 같다. 따라서, 자세한 설명은 생략하도록 한다. 여기서도 암호화 모드는 ECB 를 사용했다.

B. Hash

i. SHA256, SHA384, SHA512

```
if hash_type == "SHA256":
    print(hashlib.sha256(target_text).hexdigest())
    break
elif hash_type == "SHA384":
    print(hashlib.sha384(target_text).hexdigest())
    break
elif hash_type == "SHA512":
    print(hashlib.sha512(target_text).hexdigest())
    break
else:
    print("Wrong hash type! : " + hash_type)
```

hash 함수는 모두 hashlib 에서 제공하는 함수를 이용했다. sha256, sha384, sha512 모두 모듈만 다르게 하고, 같은 함수인 hexdigest 를 사용해서 주어진 plaintext 의 해시값을 구하도록 했다.

C. 비대칭 키 암호화

i. RSA 방식 암호화

```
def process_RSA(text):
    length = None
    encrypted = None
    decrypted = None
    rsa = None
    while True:
        try:
            while True:
                try:
                    length = int(input("key length(must be X*256, and greater or equal to 1024): "))
                except ValueError:
                    continue
                if length % 256 == 0 and length >= 1024:
                    break
            key = RSA.generate(length)
            rsa = PKCS1_OAEP.new(key)
            encrypted = rsa.encrypt(text)
            decrypted = rsa.decrypt(encrypted)
            decrypted = decrypted.decode('utf-8')
            break
        except ValueError:
            print("key length is not long enough to encrypt plain text, input larger number")
            continue

    print("encrypted: "+ str(encrypted))
    print("decrypted: "+decrypted)
```

pycryptodome 에서 RSA 를 이용한 암호화에 사용되는 모듈은 PKCS1_OAEP 이다. 일단 key 는 256 의 배수이며, 1024 보다 큰 수를 입력받도록 강제했다. 단, 이 모듈의 특성상, key 의 길이에 따라 암호화 가능한 plaintext 의 길이가 제한되었는데, 만약 key 가 너무 짧다면 ValueError 를 throw 한다. 따라서 ValueError 가 발생하면 더 큰 수를 입력하라는 메시지를 출력하고, 처음부터 과정을 다시 시작했다.

3. 실행

A. 제한 조건

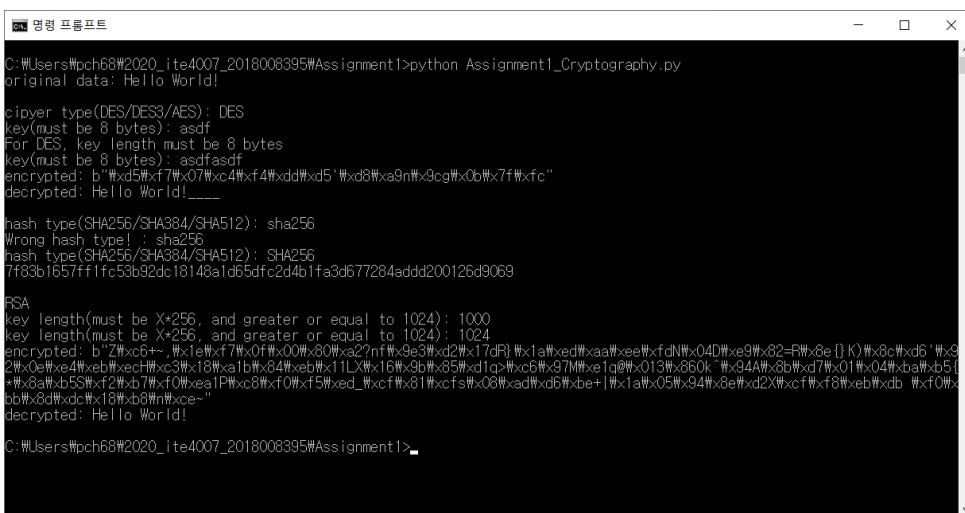
i. key 와 plaintext 입력 시의 제한

모든 입력은 utf-8 범위 내에서 해야 한다. utf-8 로 인코딩할 수 없는 문자열은 입력해서는 안된다. 이외의 조건은 모두 예외처리가 되어있다. 모든 예외 처리 방식은 단순한 재입력으로, 적절한 입력이 들어올 때까지 반복해서 입력을 받게 되어있다.

key 는 암호화 방식에 따라서 입력 창에 제한 조건이 명시되며, 이에 맞지 않는 입력을 하면 재입력을 받는다. plaintext 는 별다른 제한 조건이 없으며, 암호화 방식에 따라 복호화했을 때 padding 이 추가되어 있을 수 있다.

B. 실행 스크린샷

i. DES + SHA256 + RSA



```
C:\Users\pch68\2020_ite4007_2018008395\Assignment1>python Assignment1_Cryptography.py
original data: Hello World!

cipher type(DES/DES3/AES): DES
key(must be 8 bytes): asdf
For DES, key length must be 8 bytes
key(must be 8 bytes): asdfasdf
encrypted: b'\xd5\xf7\x07\x04\x0f\x04\x0d\x05'\xd8\xa9\x09\x0c\x0b\x07\x0f\x0c'
decrypted: Hello World!

hash type(SHA256/SHA384/SHA512): sha256
Wrong hash type! : sha256
hash type(SHA256/SHA384/SHA512): SHA256
7f83b1657ff1fc53b92dc18148a1d65d6c2d4b1fa3d677284add200126d9069

RSA
key length(must be X*256, and greater or equal to 1024): 1000
key length(must be X*256, and greater or equal to 1024): 1024
encrypted: b'Z\x06~\x1e\x0f\x00\x80\x02?nf\x9e3\xd2\x17dF}\x1a\xed\xaa\xee\xfd\x04D\x09\x82=F\x8e{}K)\x8c\xd6'\x92\x0e\x04\xeb\xec\x03\x18\x01b\x84\xeb\x11L\x16\x9b\x85\x0d1q\x06\x97M\x01g\x013\x860k"\x94A\x8b\x07\x01\x04\xba\x05(\x08a\x05S\x02\x07\x0f0\x01P\x08\x0f0\x05\x0d_\x0c\x081\x0c\x08\x06\x0d\x06\x0e+\x1a\x05\x94\x0e\x02\x0c\x0f\x08\x0b\x0b\x0f0\x0b\x0d\x0c\x18\x08n\x0e~"
decrypted: Hello World!

C:\Users\pch68\2020_ite4007_2018008395\Assignment1>
```

ii. DES3 + SHA384 + RSA



```
C:\Users\pch68\2020_ite4007_2018008395\Assignment1>python Assignment1_Cryptography.py
original data: 암호화 과제입니다.

cipher type(DES/DES3/AES): DES3
key(must be 16 or 24 bytes): 1234qwerasdf
key(must be 16 or 24 bytes): 1234qwer
key(must be 16 or 24 bytes): 1234qwerasdfzxcv
encrypted: b'\xe5\xe2\x04\x04\x07f|\xa6\x09\x08p']\xde~\xb9\x81\xaf\x06(\xc6\x03V\x08\x06\x09\x03\x07\x02\x04\x07\x0d7\x0b
decrypted: 암호화 과제입니다.

hash type(SHA256/SHA384/SHA512): SHA384
7f195952915ea8cf4e256d6b1f1c02946f4ba92a32400419a82a20c5c9e8b43525b4700cb9ccd0028468b6619c5c17a87

RSA
key length(must be X*256, and greater or equal to 1024): 2048
encrypted: b'\x81\xbe\x85\x0d\x08\x06P\x06_J\x18E\x7f\x0e\x1a\x92\x0c11\x06\x08\x04\x0d\x07\x0c\x09\x0b7Md\x08\x17\x0b=\x00\x08ab#\x0d\xdf\x02\x0af\x05\x087M\x09a6\x01aW\x02)\x0c1pd\x06\x09\x0d01\x0e6\x0a\x0830\x061#\x0a1\x0a0!\x0b3\x07\x03\x03\x010H\x05\x06>\x08b]]<\x17\x02\x0f2\x0abu\x0f9.M\x0f0h\x0e\x0c9\x0d\x0b\x1b1\x02\x01a\x04\x09\x03)\x013\x0c6f\x0f0\x0c\x03\x0c\x0e=\x02\x0c\x0b_\x07\x07\x0d\x061\x08\x0d\x0c2c\x0fa>Ca8F\x0d8n\x0e1\x08e\x12\x0b7Y\x0f9d\x0c\x013\x0c0c\x09Fy\x0a7.T\x12\x18\x052\x05w\x0c\x01\x0cd\x09121\x0eeV\x03\x09\x09\x0eq0\x044\x12)\x12\x0e\x01f)\x0c\x0da\x0b8\x0f\x0e\x09d\x09;\x0ab?\x0faF\x0c6\x06\x14817\x0ea\x04!\x0f0W\x0c2k\x091\x0b9\x0f0]\x0f6\x14\x08P\x0dbd+1\x0b5\x087gW\x0d1C\x0a58\x05\x0d\x02\x0ef\x0a7\x01a\x15_0\x02Pv\x0dc1\x09e\x0b6V\x0a0g\x0e4\x085'
decrypted: 암호화 과제입니다.

C:\Users\pch68\2020_ite4007_2018008395\Assignment1>
```

[illegible]

```
C:\Users\pch68\2020_ite4007_2018008395\Assignment1>python Assignment1_Cryptography.py  
original data: abcdabcdabcdabcdabcdabcdabcdabcdcdqwerqwerqwerqwerqwerqwerqwerqwerzxcvzxcvzxcvzxcvzxcv  
  
cipyser type(DS/DES3/AES): AES  
key(must be 16 or 24 or 32 bytes): 1234567887654321  
encrypted: b'85wx14wxfawxb7wx82twe5wx11wdx2Fwx8etxcfwxedwxc385wx14wxfwb7wx82twe5wx11wdx2Fwx8etxcfwxedwxc3Wx13Wcx5V  
wxcctVUjdxhdwx4dwwafwxbew9cyfwb4Lwxf5Wxadwxe7_jir=wf2w1Cwdfcfw99WxeDLwxf5wa4Wxe7_jir=wxf2W1Cxdfcfw98WxebD#wx  
d0WxeM#Wyc1WxcBwx99Wxdff-6Wxb[F#T]Wxf7WxdWWWwxb8Wxbdwxdfwxbfwx84Wcx7wx2wXocwx14WxdjdWxe4Wx99'  
decrypted: abcdabcdabcdabcdabcdabcdabcdabcdcdqwerqwerqwerqwerqwerqwerqwerqwerzxcvzxcvzxcvzxcvzxcv_____  
  
hash type(SHA256/SHA384/SHA512): SHA512  
9f40b3ecf00301b8e804ef91d3887814e0c7ac9748d9748287fb26505df09671649a394926cb52ef4ac5982ffd0e7f4dea53f930993ccdd54c02afa637a4  
87fa6660  
  
RSA  
key length(must be X+256, and greater or equal to 1024): 1024  
key length is not long enough to encrypt plain text, input larger number  
key length(must be X+256, and greater or equal to 1024): 1280  
encrypted: b'fww0fx12wOxc7f1fwf7WJwxcch_wxf2w11Wxd47w87Ww82cx14Wxbcdw8cxw94Wx1a_Wx08Wxd5 qWw82Wxa9Wx94Wxb52Wxe7Wx8  
25Wx02Wx92Wxf0Wxfw1e90Wxc2=Wxf6Wx12WxadWxfewxc8W=wx3Wxc2Wb1Wt1Ww10Wxd0Wxe3WcxdeQWxebWx95l1Wxa0Wx8cx1WxdA0Wxf4Wx  
15A';Wxb8kWxc5WxbexWcx7Wx80'Wx9e(Wxe4Wx17Wxb2Ww82Wxc8Wxe0'0+Wx07QWxc1sWx8fWxabWx83Wx8fWxfaWxa0Wxa9Wxc6Wx97aWixf0WxdeH  
xda'Wxa0Wx06wS/LWx8eWx8eWx0eWxielWx8aWxbcw8e9WxfaiWxb3'Wxe4Wx1x1bWx07Wxd5Wxdw76Wx93WxaaWxf0Wxf0WxiEWx12Wxe8Wx99Wvx36'  
decrypted: abcdabcdabcdabcdabcdabcdabcdabcdcdqwerqwerqwerqwerqwerqwerqwerqwerzxcvzxcvzxcvzxcvzxcv  
  
C:\Users\pch68\2020_ite4007_2018008395\Assignment1>
```

RSA 부분에서 올바른 key length 를 넣었지만, 그 크기가 충분하지 않아 재입력 메시지를 띄웠다. 더 큰 수인 1280 을 입력하니, 제대로 동작하는 것을 볼 수 있다.