Assignment1 Report

2018068622 임문경

**개요**

IDE : Jupyter notebook

language : Python3

구현 :

1. 사용자로부터 plaintext를 입력받는다.
2. Symmetric Key Cryptography : DES/DES3/AES 중 사용자가 선택하여 진행
3. Hash : SHA256/SHA384/SHA512 중 사용자가 선택하여 진행
4. Asymmetric Key Cryptography : RSA로 진행(사용자로부터 key size를 입력받는다.)

**명시사항**

1. RSA키 길이에 따른 입력 제한 : 최대 입력 문자 개수 : key length/8 – 42
2. RSA키 길이 : 최소 길이 1280, 256의 배수, 권장 길이 : 2048

**소스코드**

from Crypto.Cipher import DES

from Crypto.Random import get\_random\_bytes

from Crypto.Util.Padding import pad

from Crypto.Cipher import AES

from Crypto.Cipher import DES

from Crypto.Cipher import DES3

# input planetext

print("data : " ,end='')

data = input()

print()

**# symmetric key cryptography : Choose cryptography out of the DES/DES3/AES**

print("Cipher type(DES/DES3/AES):",end='')

cipher\_type = input()

if(cipher\_type == 'AES'): # if AES,

key = get\_random\_bytes(8) # key size = 16

cipher = AES.new(key,AES.MODE\_ECB) # use ECB\_mode

padded\_data = pad(data.encode(),16) # padding

print("key(16) : ",end=''); print(key) # output key value

elif(cipher\_type == 'DES'): # if DES,

key = get\_random\_bytes(8) # key size = 8

cipher = DES.new(key,DES.MODE\_ECB) # use ECB\_mode

padded\_data = pad(data.encode(),8) # padding

print("key(8) : ",end=''); print(key) # output key value

elif(cipher\_type == 'DES3'): # if DES3

key = get\_random\_bytes(24) # key size = 24

cipher = DES3.new(key,DES3.MODE\_ECB) # use ECB\_mode

padded\_data = pad(data.encode(),8) # padding

print("key(24) : ",end=''); print(key) # output key value

# encrypting padded data by using selected cryptography and mode

encrypted\_data = cipher.encrypt(padded\_data)

# output encrypted data

print("encrypted :",end=''); print(encrypted\_data)

# decrypting encrypted data by using selected cryptography

decrypted\_data = cipher.decrypt(encrypted\_data)

# output ecrypted data(=plane text)

print("decrypted :",end='');print(decrypted\_data.decode())

print()

import hashlib

**# Hashing algorithm : Choose Hashing algorithm out of the SHA256/SHA384/SHA512**

print("hash type(SHA256/SHA384/SHA512): ",end='')

hash\_type = input()

m = hashlib.new(hash\_type,data.encode()) # m is collector of data(=text) to be hashed

hashed\_data = m.digest() # m is hashed

print("hashed:",end=''); print(hashed\_data) # output hashed data

print()

from Crypto.PublicKey import RSA

from Crypto.Cipher import PKCS1\_OAEP

**# Asymmetric Key Cryptography : RSA**

print("RSA")

# choosing key size that is multiple of 256, and minimum key size is 1280

print("key length(x256 , at least 1280, but 2048 is recommended):",end='')

key\_size = int(input()) # input key size

## maximum text length : key length/8 - 42 ##

key = RSA.generate(key\_size) # generating key pair(private key,public key)

private\_key = key # private key

public\_key = private\_key.publickey() # public key

# generating encryptor about private key(public key is used for encrypting

encryptor = PKCS1\_OAEP.new(public\_key))

# generating encryptor about public key(private key is used for decrypting)

decryptor = PKCS1\_OAEP.new(private\_key)

# encrypting data using encryptor

encrypted\_data = encryptor.encrypt(data.encode())

# output encrypted data

print("encrypted: ",end=''); print(encrypted\_data)

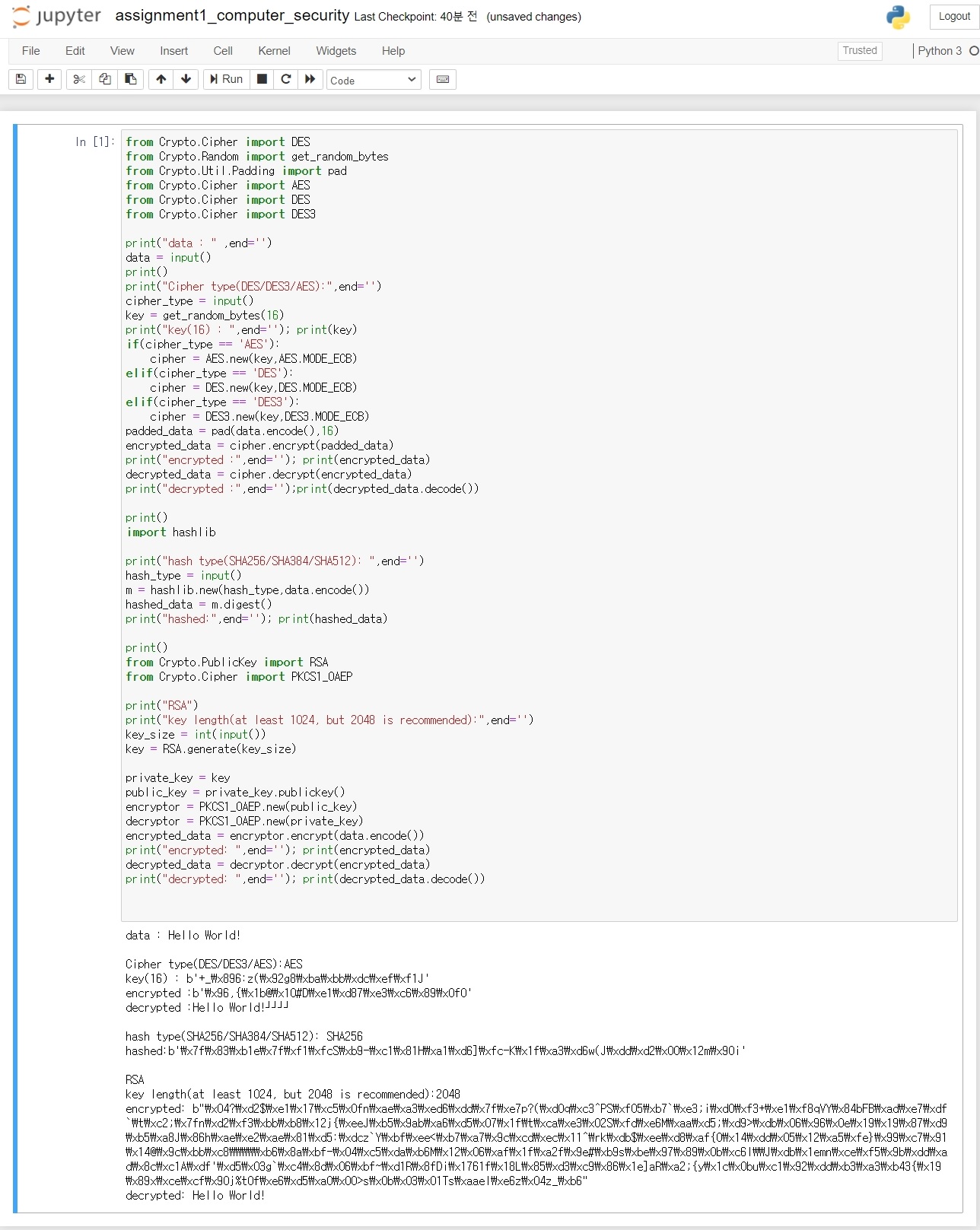
# decrypting ecrypted data using decryptor

decrypted\_data = decryptor.decrypt(encrypted\_data)

# output decrypted data(=plane text)

print("decrypted: ",end=''); print(decrypted\_data.decode())

**실행 화면**

****