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

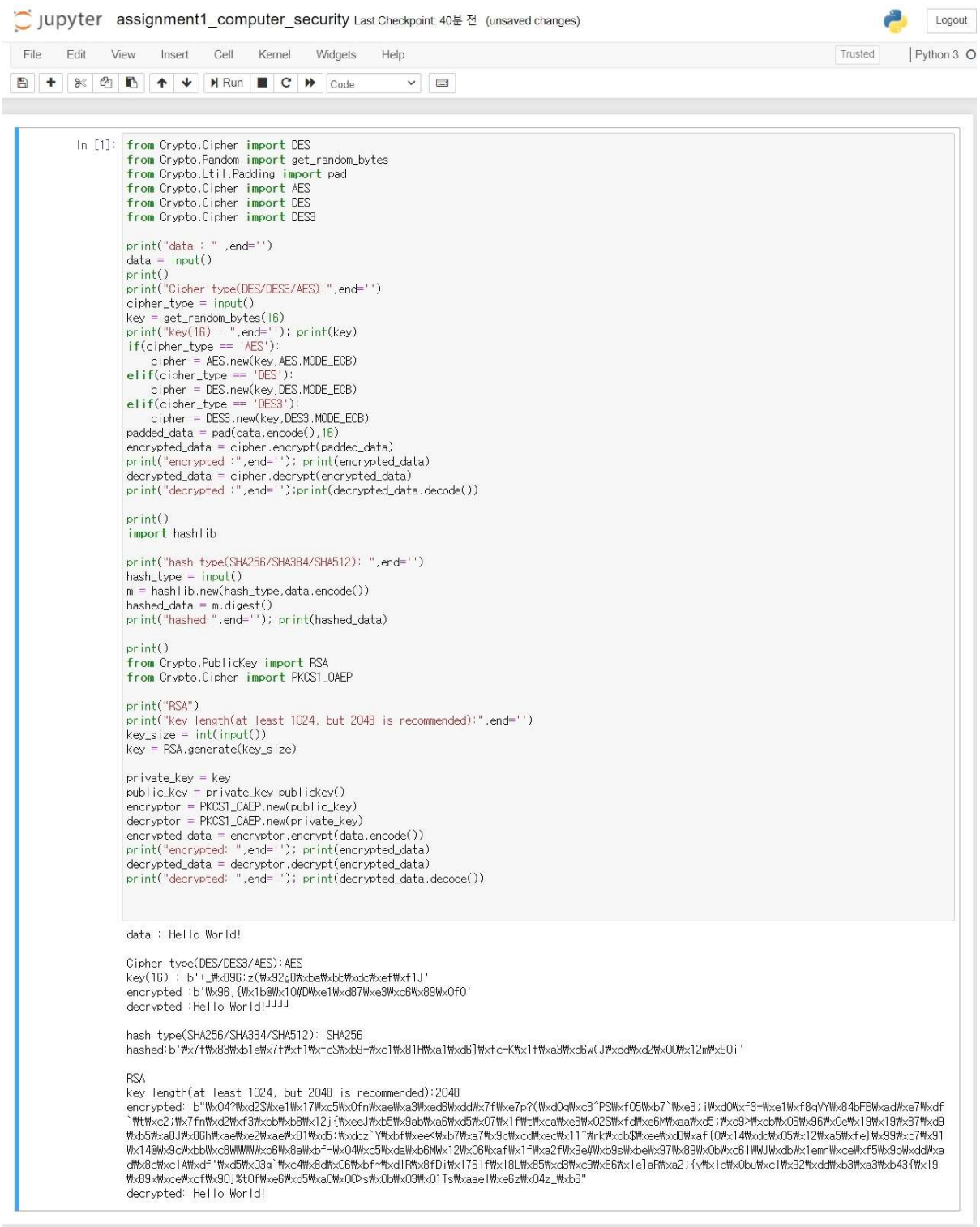
소스코드

```
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



jupyter assignment1_computer_security Last Checkpoint: 40분 전 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

In [1]:

```
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

print("data : " ,end='')
data = input()
print()
print("Cipher type(DES/DES3/AES):",end='')
cipher_type = input()
key = get_random_bytes(16)
print("key(16) : ",end=''); print(key)
if(cipher_type == 'AES'):
    cipher = AES.new(key,AES.MODE_ECB)
elif(cipher_type == 'DES'):
    cipher = DES.new(key,DES.MODE_ECB)
elif(cipher_type == 'DES3'):
    cipher = DES3.new(key,DES3.MODE_ECB)
padded_data = pad(data.encode(),16)
encrypted_data = cipher.encrypt(padded_data)
print("encrypted :",end=''); print(encrypted_data)
decrypted_data = cipher.decrypt(encrypted_data)
print("decrypted :",end='');print(decrypted_data.decode())

print()
import hashlib

print("hash type(SHA256/SHA384/SHA512): ",end='')
hash_type = input()
m = hashlib.new(hash_type,data.encode())
hashed_data = m.digest()
print("hashed:",end=''); print(hashed_data)

print()
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP

print("RSA")
print("key length(at least 1024, but 2048 is recommended):",end='')
key_size = int(input())
key = RSA.generate(key_size)

private_key = key
public_key = private_key.publickey()
encryptor = PKCS1_OAEP.new(public_key)
decryptor = PKCS1_OAEP.new(private_key)
encrypted_data = encryptor.encrypt(data.encode())
print("encrypted: ",end=''); print(encrypted_data)
decrypted_data = decryptor.decrypt(encrypted_data)
print("decrypted: ",end=''); print(decrypted_data.decode())
```

```
data : Hello World!

Cipher type(DES/DES3/AES):AES
key(16) : b'+_₩x896:z(₩x92g8₩xba₩xbb₩xdc₩xef₩xf1J'
encrypted :b'₩x96,{₩x1b@₩x1Q#D₩xe1₩xd87₩xe3₩xc6₩x89₩x0f0'
decrypted :Hello World!JJJJ

hash type(SHA256/SHA384/SHA512): SHA256
hashed:b'₩x7f₩x83₩xb1e₩x7f₩xf1₩xfcS₩xb9-₩xc1₩x81H₩xa1₩xd6]₩xfc-K₩x1f₩xa3₩xd6w(J₩xd₩xd2₩x00₩x12m₩x90i'

RSA
key length(at least 1024, but 2048 is recommended):2048
encrypted: b"₩x047₩xd2$₩xe1₩x17₩xc5₩x0fn₩xae₩xa3₩xed6₩xdd₩x7f₩xe7p?(₩xd0q₩xc3^PS₩xf05₩xb7`₩xe3;i₩xd0₩xf3+₩xe1₩xf8qVY₩x84bFB₩xad₩xe7₩xdf
`₩t₩xc2;₩x7fn₩xd2₩xf3₩xbb₩xb8₩x12j{₩xeeJ₩xb5₩x9ab₩xa6₩xd5₩x07₩x1f₩t₩xca₩xe3₩x02S₩xfd₩xe6M₩xaa₩xd5;₩xd9>₩xdb₩x06₩x96₩x0e₩x19₩x19₩x87₩xd9
₩xb5₩xa8J₩x86h₩xae₩xe2₩xae₩x81₩xd5:₩xdcz`Y₩xbf₩xee<₩xb7₩xa7₩x9c₩xcd₩xec₩x11^₩rk₩xdb$₩xee₩xd8₩xaf{0₩x14₩xdd₩x05₩x12₩xa5₩xfe}₩x99₩xc7₩x91
₩x140₩x9c₩xbb₩xc8₩₩₩₩₩xb6₩x8a₩xbf-₩x04₩xc5₩xda₩xb6M₩x12₩x06₩xaf₩x1f₩xa2f₩x9e#₩xb9s₩xbe₩x97₩x89₩x0b₩xc6l₩₩J₩xdb₩x1emn₩xce₩xf5₩x9b₩xdd₩xa
d₩x8c₩xc1A₩xdf'₩xd5₩x03g`₩xc4₩x8d₩x06₩xbf~₩xd1R₩x8fDi₩x1761f₩x18L₩x85₩xd3₩xc9₩x86₩x1e]aF₩xa2;{y₩x1c₩x0bu₩xc1₩x92₩xdd₩xb3₩xa3₩xb43{₩x19
₩x89x₩xce₩xcf₩x90j%t0f₩xe6₩xd5₩xa0₩x00>s₩x0b₩x03₩x01Ts₩xaael₩xe6z₩x04z_₩xb6"
decrypted: Hello World!
```