블록체인 실습 – AES 암호 알고리즘 실습(CBC모드 적용)

■ 코드 실행해보기

```
from Crypto.Cipher import AES
from Crypto import Random
import numpy as np
import hashlib as h

#원문
plainText = 'AES 알고리즘 실습하기'
print("₩n plain text:", plainText)

#256bit의 key 생성
key = Random.get_random_bytes(32)
print("₩n secret key : ", key)

#AES 블록크기 배수에 맞게 패딩
encodedtext = plainText.encode()
n = len(encodedtext)
if (n%16) == 0:
    PlainText = encodedtext
if (n%16) !=0:
    n = n+16 - (n%16)
    PlainText = encodedtext.ljust(n, b'₩0')

#CBC모드에서 사용할 초기 벡터 생성
iv = Random.new().read(AES.block_size)

#암호화 하기
aes = AES.new(key, AES.MODE_CBC, iv)
cipherText = aes.encrypt(PlainText)
print("₩n cipher text: ", cipherText)

#복호화 하기
aes = AES.new(key, AES.MODE_CBC, iv)
checkplaintext = aes.decrypt(cipherText)
print("₩n after decryption :", checkplaintext)

#디코딩하여 원문 확인
checkplaintext = checkplaintext.decode()
print("₩n 해독문:", checkplaintext)
```

- **Random.get_random_bytes(N)**

: N길이의 random byte string 값을 return

: 256 bit(= 32 byte)의 key를 생성하므로 32입력

- **encodedtext = plainText.encode(encoding='UTF-8')**

: plainText를 인코딩 (유니코드로 저장하기 위함)

: default 값은 UTF-8

블록체인 실습 – AES 암호 알고리즘 실습(CBC모드 적용)

■ 코드 실행해보기

```
from Crypto.Cipher import AES
from Crypto import Random
import numpy as np
import hashlib as h

#원문
plainText = 'AES 알고리즘 실습하기'
print("₩n plain text:", plainText)

#256bit의 key 생성
key = Random.get_random_bytes(32)
print("₩n secret key : ", key)

#AES 블록크기 배수에 맞게 패딩
encodedtext = plainText.encode()
n = len(encodedtext)
if (n%16) == 0:
    PlainText = encodedtext
if (n%16) !=0:
    n = n+16 - (n%16)
    PlainText = encodedtext.ljust(n, b'₩0')

#CBC모드에서 사용할 초기 벡터 생성
iv = Random.new().read(AES.block_size)

#암호화 하기
aes = AES.new(key, AES.MODE_CBC, iv)
cipherText = aes.encrypt(PlainText)
print("₩n cipher text: ", cipherText)

#복호화 하기
aes = AES.new(key, AES.MODE_CBC, iv)
checkplaintext = aes.decrypt(cipherText)
print("₩n after decryption :", checkplaintext)

#디코딩하여 원문 확인
checkplaintext = checkplaintext.decode()
print("₩n 해독문:", checkplaintext)
```

- **len(str)**

  : 문자열의 길이를 구함(byte 단위)

  : 코드에서는 encodedtext의 길이를 n에 저장

- **If (n%16) == 0: PlainText = encodedtext**

  : encodedtext의 길이가 16byte의 배수라면 인코딩한 평문을 PlainText에 대입함

- **If (n%16) != 0: n = n+16 – (n%16)**

  : encodedtext의 길이가 16byte의 배수가 아니라면 16을 더하고 나머지 값을 빼서 16byte의 배수가 되도록 n 값을 조정

- **str.ljust(width, filchar)**

  : str의 길이가 width(int 형)될때까지 왼쪽부터 filchar(byte 형)을 채움

  : 코드에서 n, b'₩0' 입력(b를 붙이면 byte로 변환)

블록체인 실습 – AES 암호 알고리즘 실습(CBC모드 적용)

■ 코드 실행해보기

```
from Crypto.Cipher import AES
from Crypto import Random
import numpy as np
import hashlib as h

#원문
plainText = 'AES 알고리즘 실습하기'
print("₩n plain text:", plainText)

#256bit의 key 생성
key = Random.get_random_bytes(32)
print("₩n secret key : ", key)

#AES 블록크기 배수에 맞게 패딩
encodedtext = plainText.encode()
n = len(encodedtext)
if (n%16) == 0:
    PlainText = encodedtext
if (n%16) !=0:
    n = n+16 - (n%16)
    PlainText = encodedtext.ljust(n, b'₩0')

#CBC모드에서 사용할 초기 벡터 생성
iv = Random.new().read(AES.block_size)

#암호화 하기
aes = AES.new(key, AES.MODE_CBC, iv)
cipherText = aes.encrypt(PlainText)
print("₩n cipher text: ", cipherText)

#복호화 하기
aes = AES.new(key, AES.MODE_CBC, iv)
checkplaintext = aes.decrypt(cipherText)
print("₩n after decryption :", checkplaintext)

#디코딩하여 원문 확인
checkplaintext = checkplaintext.decode()
print("₩n 해독문:", checkplaintext)
```

- **Random.new().read(AES.block_size)**
  : AES의 블록 사이즈와 동일한 랜덤한 초기 벡터를 생성
- **Crypto.Cipher.AES.new(key, mode, *args, **kwargs)**

:새로운 AES 암호 모듈을 생성

:key에는 대칭키, mode는 사용하는 운영모드 종류, *args, **kwargs는 사용하는 운영모드마다 다름

:이 코드에선 CBC모드를 사용하여 초기 벡터도 필요

블록체인 실습 – AES 암호 알고리즘 실습(CBC모드 적용)

■ 코드 실행해보기

```
from Crypto.Cipher import AES
from Crypto import Random
import numpy as np
import hashlib as h

#원문
plainText = 'AES 알고리즘 실습하기'
print("₩n plain text:", plainText)

#256bit의 key 생성
key = Random.get_random_bytes(32)
print("₩n secret key : ", key)

#AES 블록크기 배수에 맞게 패딩
encodedtext = plainText.encode()
n = len(encodedtext)
if (n%16) == 0:
    PlainText = encodedtext
if (n%16) !=0:
    n = n+16 - (n%16)
    PlainText = encodedtext.ljust(n, b'₩0')

#CBC모드에서 사용할 초기 벡터 생성
iv = Random.new().read(AES.block_size)

#암호화 하기
aes = AES.new(key, AES.MODE_CBC, iv)
cipherText = aes.encrypt(PlainText)
print("₩n cipher text: ", cipherText)

#복호화 하기
aes = AES.new(key, AES.MODE_CBC, iv)
checkplaintext = aes.decrypt(cipherText)
print("₩n after decryption :", checkplaintext)

#디코딩하여 원문 확인
checkplaintext = checkplaintext.decode()
print("₩n 해독문:", checkplaintext)
```

- **aes.encrypt(PlainText)**

  :평문을 AES 암호 알고리즘을 이용하여 암호화하여 암호문 생성

- **aes.decrpt(cipherText)**

  :암호문을 AES 암호 알고리즘을 이용하여 복호화하여 평문을 복구

- **checkplaintext.decode()**

  :checkplaintext(평문)를 보기 편하게 디코딩함

블록체인 실습 – AES 암호 알고리즘 실습(CBC모드 적용)

■ 출력 예시

```
plain text: AES 알고리즘 실습하기
secret key :  b'₩xcd₩xad₩x1b₩xb0₩xe3₩x021₩x0b5₩xb3$₩xe8(₩x84₩x1c₩xb2₩n1.7₩xdf₩xb3₩₩xab₩x1f₩xd5₩xd3₩xe6₩x89₩x91₩xae₩xc8'
cipher text:  b'₩xb0₩r₩x14₩xe6n₩xaa₩xd9₩x0f+₩xb8₩xa6y₩xa7₩x9fQ₩x1a,0₩x10₩xa4₩xadU₩xa4₩x9c₩xd7l₩xdc₩xa5w₩xf9:s'
after decryption : b'AES ₩xec₩x95₩x8c₩xea₩xb3₩xa0₩xeb₩xa6₩xac₩xec₩xa6₩x98 ₩xec₩x8b₩xa4₩xec₩x8a₩xb5₩xed₩x95₩x98₩xea₩xb8₩xb0₩x00₩x00₩x00'
해독문: AES 알고리즘 실습하기
>>>
```

블록체인 실습 – AES 암호 알고리즘 실습(CBC모드 적용)

■ 코드 실행해보기(padding 모듈 사용)

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


message = '이것은 평문입니다'
print("₩n 평문:", message)
encodedtext = message.encode()

key = get_random_bytes(32)
iv = get_random_bytes(16)

aes = AES.new(key, AES.MODE_CBC, iv)
ciphertext = aes.encrypt(pad(encodedtext, 16))

print("₩n 암호문:", ciphertext)
aes= AES.new(key, AES.MODE_CBC, iv)
plaintext = unpad(aes.decrypt(ciphertext), 16)
plaintext = plaintext.decode()

print("₩n 해독문:", plaintext)
```

- **From Crypto.Util.Padding import pad, unpad**

  : Crypto에서 제공하는 Padding 모듈 사용

- **Pad(text, blocksize, style = 'pkcs7')**

  :text에는 패딩이 필요한 메시지, blocksize는 블록의 길이, style은 사용하는 패딩 종류를 입력

  : style은 pkcs7이 디폴트 값

  : unpad(패딩 제거)도 마찬가지 방법으로 사용

블록체인 실습 – AES 암호 알고리즘 실습(CBC모드 적용)

■ 출력 예시(padding 모듈 사용)

```
평문: 이것은 평문입니다
암호문: b'>J₩xeaJy₩x13₩xb9^₩x14₩x00a₩xa66wm₩xd2wel₩x98₩xebZ₩xbbz₩xc5:₩x1a!g₩xb7₩x07
₩xaf'
해독문: 이것은 평문입니다
```

블록체인 실습 – SHA256 해쉬 알고리즘

■ 코드 실행해보기

```
from Crypto.Hash import MD5, RIPEMD, SHA, SHA256

msg = input("Message : ")
msg = msg.encode()

h = SHA256.new()                    # SHA256
h.update(msg)
hashv = h.hexdigest()
print("       SHA256 : ", hashv)
print("       길이 : ", len(hashv)*4)

h1 = SHA256.new()                   # Double-SHA256
h1.update(msg)
h2 = SHA256.new()
h2.update((h1.hexdigest()).encode())
hashv = h2.hexdigest()
print("₩nDouble-SHA256 : ", hashv)
print("       길이 : ", len(hashv)*4)
```

- **From Crypto.Hash import MD5, RIPEMD, SHA, SHA256**

  :Crypto 안 Hash 모듈에서 여러 해쉬 알고리즘을 임포트

  :여기서는 SHA256만 사용
- **Input(“msg”)**

  :msg 부분을 화면에 출력하고 입력값을 받음
- **SHA256.new(data=None)**

  :새로운 hash 모듈 생성

  :data는 해쉬할 메시지의 맨 첫부분 입력(옵션)
- **h.update(msg)**

  :msg의 해쉬를 진행

블록체인 실습 – sha256 해쉬 알고리즘

■ 코드 실행해보기

```
from Crypto.Hash import MD5, RIPEMD, SHA, SHA256

msg = input("Message : ")
msg = msg.encode()

h = SHA256.new()                    # SHA256
h.update(msg)
hashv = h.hexdigest()
print("       SHA256 : ", hashv)
print("       길이 : ", len(hashv)*4)

h1 = SHA256.new()                   # Double-SHA256
h1.update(msg)
h2 = SHA256.new()
h2.update((h1.hexdigest()).encode())
hashv = h2.hexdigest()
print("₩nDouble-SHA256 : ", hashv)
print("       길이 : ", len(hashv)*4)
```

- **h.hexdigest()**
  : 이전에 생성된 메시지의 해쉬값을 출력가능한 string으로 반환
- **h2.update((h1.hexdigest()).encode())**
  : 이전에 생성된 메시지의 해쉬값을 인코딩한 후 다시 해쉬를 함
  : Double-SHA256
- **print(“msg”, hashv)**
  :msg내용을 출력하고 hashv의 값을 출력

블록체인 실습 – SHA256 해쉬 알고리즘

■ 출력 예시

```
Message : Hashvalue 계산하기
        SHA256 :  a0564ebe50a9fd49eaae0dc4fcad82c5cf910ffe684a4eaa7cea4a055457688f
        길이 :  256

Double-SHA256 :  c560505e6c30e6d48f6e1c94a96d98c641bab80aa37f73f157e68805553f951d
        길이 :  256
```

블록체인 실습 – RSA 암호 알고리즘 실습

■ 코드 작성

```
from Crypto.Cipher import PKCS1_OAEP
from Crypto.PublicKey import RSA
from Crypto import Random

# Private key 생성
privkey = RSA.generate(2048)
# 생성한 private key와 쌍인 public key 생성
pubKey = privkey.publickey()

print("₩nprivate key : ", privkey.exportKey().decode())
print("₩npublic key : ", pubKey.exportKey().decode())

#사용하는 p, q, e, d 값 확인
print("₩n p = ", privkey.p)
print("₩n q = ", privkey.q)
print("₩n e = ", privkey.e)
print("₩n d = ", privkey.d)
```

- **From Crypto.Cipher import PKCS1_OAEP**
  : PKCS1_OAEP를 사용하여 RSA 암호화, 복호화를 진행
- **From Crypto.PublicKey import RSA**
  : Crypto안의 Publickey 모듈 중 RSA 사용
- **RSA.generate(2048)**
  : RSA 알고리즘에 사용할 개인키를 생성
- **privkey.publickey()**
  : 이전에 생성한 개인키와 쌍인 공개키를 생성

블록체인 실습 – RSA 암호 알고리즘 실습

■ 코드 작성

```
from Crypto.Cipher import PKCS1_OAEP
from Crypto.PublicKey import RSA
from Crypto import Random

# Private key 생성
privkey = RSA.generate(2048)
# 생성한 private key와 쌍인 public key 생성
pubKey = privkey.publickey()

print("₩nprivate key : ", privkey.exportKey().decode())
print("₩npublic key : ", pubKey.exportKey().decode())

#사용하는 p, q, e, d 값 확인
print("₩n p = ", privkey.p)
print("₩n q = ", privkey.q)
print("₩n e = ", privkey.e)
print("₩n d = ", privkey.d)
```

- **privkey.exportKey().decode()**
  : 연속적인 키 값 출력을 위해 exportKey()를 사용
  : 보기 편하게 디코딩

- **privkey.p**
- **privkey.q**
- **privkey.e**
- **privkey.d**
  : 각각 RSA 알고리즘에서 사용하는 해당 변수를 출력

블록체인 실습 – RSA 암호 알고리즘 실습

■ 코드 작성

```
# 암호화할 원문
plainText = "공개키 알고리즘 RSA 연습하기"
print("₩n")
print("원문 : ", plainText)

#공개키로 암호화하기
key = PKCS1_OAEP.new(pubKey)
cipherText = key.encrypt(plainText.encode())
print("₩n")
print("암호문 : ", cipherText)

# 수신자의 Private key로 암호문을 복호화하기
key = PKCS1_OAEP.new(privkey)
checkplainText = key.decrypt(cipherText)
checkplainText = checkplainText.decode()
print("₩n")
print("해독문 : ", checkplainText)
```

- **PKCS1_OAEP.new(key)**

  : 암호화 혹은 복호화를 하기 위한 암호 객체 생성

  : key 값은 암호화 할 때는 공개키, 복호화 할때는 개인키 사용

- **key.encrypt(message)**

  : 해당 메시지를 암호화(byte형으로 반환)

- **key.decrypt(ciphertext)**

  : 해당 암호문을 복호화(byte형으로 반환)

블록체인 실습 – RSA 암호 알고리즘 실습

■ 출력 예시 – 개인키,공개키

```
private key :  -----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAk3/5c17OZjQ2cRY8CGcAJMBdpeBncieodwJkx+Afjv4OJO8I
fUUgRdsjiRf3d/iOkdv54z3FlmjxMm1MG1ROb42CLQaZQX/O1+GwzLnjEhwSE7fI
3G/aE1LCWeUgI/ogNFoCEbHYXggwSBZeC1sgOEzZbue9X22dBNhcGclwweqrTyxu
eqlAUck3fd8RRGg31ElG5dGOZSHKcsGU+CGs+VU/yD7ludDQkt2dUOCUVk+iLsln
MetytlVFhRzaWfZK4unnOsnc9DtJK5eNN92ZNtCI52HNG8Pj7yLCS8NYgByX9tAU
msRAuGurUVoOuFxugGvfQXyt4+51sCIXTJRb7QIDAQABAoIBADVx206jSxFAWPun
7U+OFpq6cxRQF9uGoCWjleZEgNrcpAQcSv61heiZ6ynZG8uG09NgDA75SndiBgda
EAoVBaio8yKicfSISyzP87JypmLg9fHcH/c8Nh3rTk2pVqKKuJGPjZFgyf/kYq4O
lJZnyh9/wQdNA9Kokoz3YoPbeDZi5yavOqgkumBmMyh/rBqT2KMQjzdQOged8hxJ
jTcVFJng+lNtOenHRiTAWmNL+1g8v4CDxmZNT7GdvwOcXBiQikXSbVgW1hBdGYXh
sx8DfbEUMVQi8ABKxZTdhOoovLlVAyAl2yPFlsy3lpkDOYgYpC7T3M/cnftgwT1n
CoWoMbcCgYEAwj67pGfCOixcFbMEuXgAeOzE8GnJJAaREKByRYAEOoMjJcj96T59
W76hdECNGh3TqsC/qMVWm/X/GaWJZJbVZRoaty7SUhOdy+P3Rt26NXOLF2QP/rpY
+9daFdQ5JtWranDrSanR4zuBWmWuJ4TBwfotgcz8wxoOxr+X+hyGXCsCgYEAwmS8
lX6crQLn/lLbXxP47iaiLNxilqf4Y5il+wNVsnjEGaMQQ+4Aft7vSc/aNdp/n7or
1o9BNXk8/d/Uqfl7ix8gfrlHNBZQd2Nv/R99rKDXrtyHa/OnEvtWl6cx5GdUve6V
bcC71MFO5+gB8eq9+MaERRrVVUBQZj1OWP3wZEcCgYBCIL483FP3HL1zsqkr6bY/
vOjEf34ymw2eZWDF8du1TL3/daWTDZhqmiO9a+LxOMd9ABSyBPJrAyct4O/O6r6E
qPf/Uk1FWDWebcWUxOvO6c2jdukEp1tY97gpQ1VTumnHUm1UEqGfAgaUtcW8RgUk
OFzg4SLCvXjZp1KwqJZIgQKBgQCCuMkGVSUs9yJX3Qt2QXEJfTNbpYwqsaPlHfOn
+9MaJDoC33PlqIvs9S/G3B5qTpe6KrCmJUI9kxzfwDA7ZGiElpdwgFmQflz4HAdl
GXAisTq5HRV4E65Lhnbxnn5fkNkcrB8SYiQtaxOLuCnuq606coyqB+Rj4s7uldXY
5FNOqwKBgHtGVBpNQfLEYDZY1ThbU6r2Y722OuMEQfhweC6DobrugLuVfD1CjkbJ
qWDWc1OnnGxnbwp6Ab1kyAJBsOh45/UkL8OI8Vdt2HEaA/AaVioGOtTYtKh+dle7
ZJI+ONho+OO44JEKNsraaDZUjs3L8MNSa9zkqU1vxCMK7fSU5sO9
-----END RSA PRIVATE KEY-----

public key :  -----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAk3/5c17OZjQ2cRY8CGcA
JMBdpeBncieodwJkx+Afjv4OJO8IfUUgRdsjiRf3d/iOkdv54z3FlmjxMm1MG1RO
b42CLQaZQX/O1+GwzLnjEhwSE7fI3G/aE1LCWeUgI/ogNFoCEbHYXggwSBZeC1sg
OEzZbue9X22dBNhcGclwweqrTyxueqlAUck3fd8RRGg31ElG5dGOZSHKcsGU+CGs
+VU/yD7ludDQkt2dUOCUVk+iLsInMetytlVFhRzaWfZK4unnOsnc9DtJK5eNN92Z
NtCI52HNG8Pj7yLCS8NYgByX9tAUmsRAuGurUVoOuFxugGvfQXyt4+51sCIXTJRb
7QIDAQAB
-----END PUBLIC KEY-----
```

블록체인 실습 – RSA 암호 알고리즘 실습

■ 출력 예시 – p,q,e,d값 출력

```
p =  14081092854235766300375660656123949172268027070732409482624886410040772124883542671750311031249513599663454353829126545641522477472267018899999472110147944333888916005503760623961173035005357343045159647166410433664529513214807702805446184172007322945097365332032584476959355211021435327136586240706194436385 9

q =  15038831927517305373294478693860107947213484073218931771597404366917403375087599832719009171459988046318111642872591646314499298245700494883314868839505357903518097708867201126003237018218531798778499626936335119034442501900826365989198209393120761656943663112606826821722992908758215783029390181707269845920 9

e =  65537

d =  99695460990672220727470906797422190982559566029623477122212327395283693915154221311282702785453996113112698195379441688758289924588543616727690724652024493457603977722766877232739354779117433714775830879729354631215537644706563804082571855691615035433686510062700965022796136176815378636596831048159246637207757788162361293676144970646417017154512463459917513628192014284960272517695022095685947961912466322664747383015782339502538413828856880357484042626023313803911017007320548361610877413543538943358201282233684819517838640785443069477470508384229152397913739621776449653110240222122806796868322200375379699185 61
```

블록체인 실습 – RSA 암호 알고리즘 실습

■ 출력 예시 – 암호문, 해독문

```
원문 :  공개키 알고리즘 RSA 연습하기

암호문 :  b'₩x8bCz₩xee₩x8e₩xeePo₩x9ct₩x90₩xe1₩xf6₩x06₩xd2H₩x1c+₩x15₩x0b₩x8c₩xcd3₩x8f
₩x82h₩xc6₩xc4`₩xa3₩xd6vV{%₩x12:A₩xa2₩x81₩xb4₩xa5₩xac₩x01₩xa3₩x12)₩xf80sr#₩xe8i₩x18₩x
93₩xbdC$₩xe5₩xe2₩xf6₩xfb₩xf4Rm₩x1f₩x9cv₩xcc₩x05C₩xea.₩xfc5Qh₩xca₩x18₩x99₩x9d₩xdb₩xb3
₩xf8₩x89e₩xede₩x98₩xbd₩xbb5₩x1cr₩x1222₩xfa₩xea₩xb9₩xcc₩xac₩x04₩xd3^₩x94₩xb1₩xc6₩x9a,
h₩xff₩x14k₩x14₩x9e₩x90@r₩xd2₩x05₩xb4tK₩x9dp-₩xcd>₩₩₩xc7₩xee₩x03₩xdd₩xa2₩xc5₩xa7₩xc3₩
xf75qH₩xd6₩xff₩xfcLzi+₩xd3₩xa3₩xd5₩xc9₩x10₩x91₩xd5₩xf5Nq"1₩x8f₩xed₩xa7₩xdeBy₩x91₩xa3
₩xe2h*o₩xee₩xa2-[i₩xa6₩xd5₩xba₩xb9₩xa2₩n"₩xf9₩xb9₩x92₩xa4₩xb0₩xb0%₩xe2₩x84d`₩x1b₩xc7
₩x935₩x9b;₩xd6E₩xcc6gGOT[₩x0f₩xe2x3₩xa8gp1%d₩x0bi₩xe6₩x83Mv*₩x1d₩xd2₩xf9#[₩xc9o₩xc3₩
xaeY:₩xaaC₩x83₩xa4₩xd6₩xc61%₩x08₩xceZ₩xb9₩x03₩x10₩x18₩x1a'

해독문 :  공개키 알고리즘 RSA 연습하기
```

블록체인 실습 – 전자 서명(Digital Signature)

■ ECDSA

```
import bitcoin.main as btc

# 개인키와 공개키를 생성
d = btc.random_key()
Q = btc.privkey_to_pubkey(d)


# 서명할 메시지 입력
msg = input("write massage : ")
msg= msg.encode()

# Signature
sig = btc.ecdsa_sign(msg, d)
print("₩n signature : ",sig)

# Verification
v = btc.ecdsa_verify(msg, sig, Q)

print("₩nVerify signature")
print("₩nmessage= ", msg.decode())

if v:
    print(v)
    print("₩n* Verification success")
else:
    print("₩n* Invalid Signature")
```

- **Import bitcoin.main as btc**

  : bitcoin 라이브러리 사용

- **Btc.random_key()**

  : 비트코인에서 사용하는 개인키 생성

- **Btc.privkey_to_pubkey(privatekey)**

  : Privatekey에 개인키를 입력하고 이와 쌍인 공개키를 생성

블록체인 실습 – 전자 서명(Digital Signature)

■ ECDSA

```
import bitcoin.main as btc

# 개인키와 공개키를 생성
d = btc.random_key()
Q = btc.privkey_to_pubkey(d)


# 서명할 메시지 입력
msg = input("write massage : ")
msg= msg.encode()

# Signature
sig = btc.ecdsa_sign(msg, d)
print("₩n signature : ",sig)

# Verification
v = btc.ecdsa_verify(msg, sig, Q)

print("₩nVerify signature")
print("₩nmessage= ", msg.decode())

if v:
    print(v)
    print("₩n* Verification success")
else:
    print("₩n* Invalid Signature")
```

- **Btc.ecdsa_sign(msg, privatekey)**
  : 개인키로 메시지 msg에 서명
- **Btc.ecdsa_verify(msg, sig, publickey)**
  : 공개키를 통해 메시지 msg와 서명 sig으로 검증
  : 검증에 성공하면 True 반환

-> v가 true일 경우 verification success 출력
그렇지 않은 경우 Invalid Signature 출력

블록체인 실습 – 전자 서명(Digital Signature)

■ 출력 예시

```
write massage : 오늘 점심 메뉴는 치킨입니다.
 signature :  G9rR55iUBU1nyINBOLKFI6k8OizzBUpjJ3GAIac7TW3YXZmQmBZvATCjokWXgTBpepLqQoPwqVjCc85dNYe/Zsk=
Verify signature
message=  오늘 점심 메뉴는 치킨입니다.
True
* Verification success
```

- 서명을 서명자의 공개키로 검증하여 해당 문서가 서명자가 작성했다는 것을 증명한다.