Block Cipher Operation

分工

組員A: B10730007 楊芯蘋
組員B: B10730027 曾瑄翎

	ECB	CTR	Custom
組員A		V	V
組員B	V		V
說明	前期共同討論, 過程中互相幫忙解決問題, 最後一起整合		

建置環境 / 依賴套件

- **Pillow**
  將圖片轉為 ppm 格式
- **pycryptodome**
  使用AES進行加密
- bitstring

操作方式

Make

```
make install
make enc
```

```
make dec
```

Input & Output

加密

./enc.py {filename}.jpeg {mode}(ECB,CTR,CUSTOM)

```
./enc.py linux.jpeg ECB
./enc.py linux.jpeg CTR
./enc.py linux.jpeg CUSTOM
```

加密完成的圖片將存放在test_enc目錄

解密

./dec.py {mode}(ECB,CTR,CUSTOM)

```
./dec.py ECB
./dec.py CTR
./dec.py CUSTOM
```

解密完成的圖片將存放在test_dec目錄

程式碼解說

加密大綱

> 輸入原始圖片、模式

```
pic_file = sys.argv[1]
mode = sys.argv[2]
```

> 將原始圖片轉成ppm

```
# Convert to ppm
ppmPicture = "./input_enc.ppm"
im = Image.open(sys.argv[1])
im.save(ppmPicture)
tmpPic = "./tmp.ppm"
```

> 加密模式定義

```
> ecb
> ctr
> custom
```

解密大綱

> 輸入模式

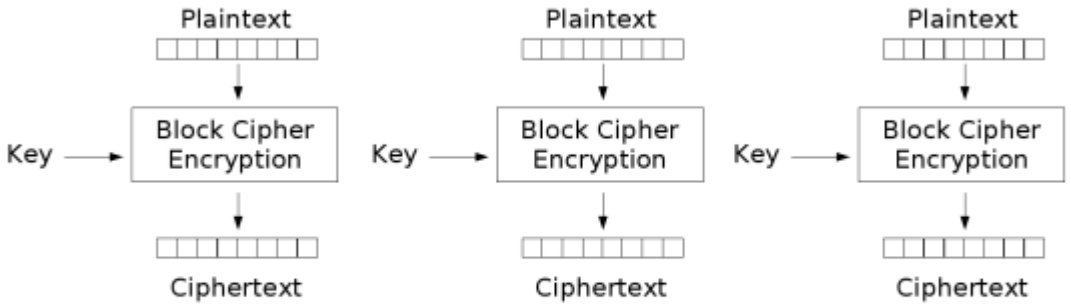
```
pic_file = sys.argv[1]
mode = sys.argv[1]
```

> 解密模式定義

```
> ecb
> ctr
> custom
```

ECB 模式

加密



Electronic Codebook (ECB) mode encryption

1. 設定輸入和輸出檔案、key、AES cipyer、Padding

```
# Read ppm as a binary file
bin_data = open(ppmPicture, 'rb').read()

t = open(tmpPic, "wb")

key = b'MORECOFFEEPLEASE'
cipher = AES.new(key, AES.MODE_ECB)

# Add null bytes
while (len(bin_data) % 16 != 0):
    bin_data += b'\x00'

t.write(bin_data)
# clear output
output = open("out_ecb_enc.ppm", "wb")
output.close()

output = open("out_ecb_enc.ppm", 'ab')
line_count = 0
```

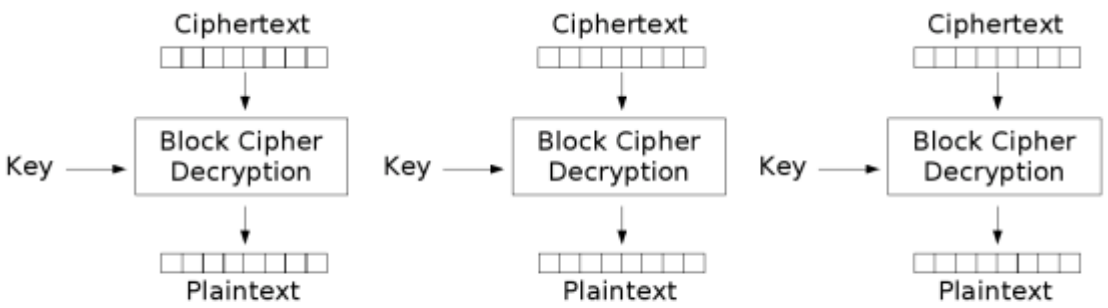
2. 對每塊明文加密，寫檔

```python
    with open(tmpPic, 'rb') as f:
        while True:
            buf = f.read(16)
            if line_count == 0:
                msg = buf
            else:
                #對每塊明文加密
                msg = cipher.encrypt(buf)
            output.write(msg)
            line_count += 1
            if not buf:
                break
```

3. 將ppm轉成jpeg，存檔

```python
    # Convert to jpeg
    Picture = "./out_ecb_enc.ppm"
    im = Image.open(Picture)
    im.save("./test_enc/ECB.jpeg")
```

解密



Electronic Codebook (ECB) mode decryption

1. 設定輸入和輸出檔案、key、AES decipyer

```python
    input = "./out_ecb_enc.ppm"

    output = open("out_ecb_dec.ppm", "wb")
    output.close()
    output = open("out_ecb_dec.ppm", 'ab')
    line_count = 0
    key = b'MORECOFFEEPLEASE'
    decipher = AES.new(key, AES.MODE_ECB)
```

2. 對每塊密文解密，寫檔

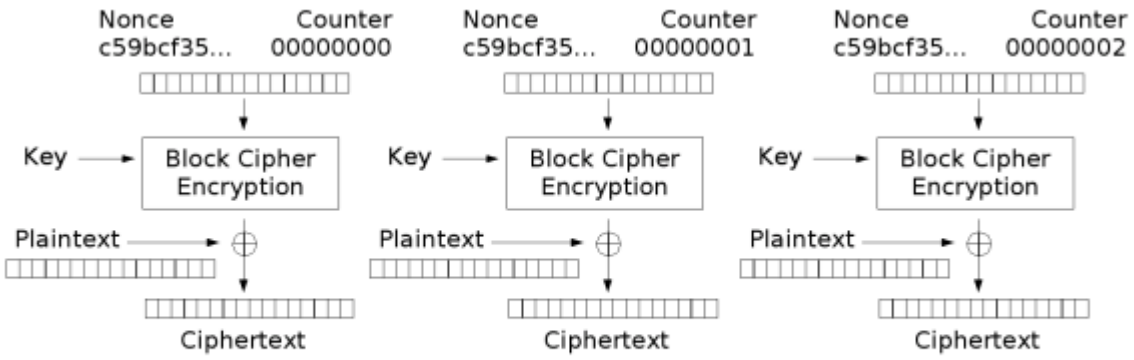
```
    with open(input, 'rb') as f:
        while True:
            buf = f.read(16)
            if line_count == 0:
                msg = buf
            else:
                #對每塊密文解密
                msg = decipher.decrypt(buf)
            output.write(msg)
            line_count += 1
            if not buf:
                break
```

3. 將ppm轉成jpeg，存檔

```
    # Convert to jpeg
    Picture = "./out_ecb_dec.ppm"
    im = Image.open(Picture)
    im.save("./test_dec/ECB.jpeg")
```

CTR 模式

加密



Counter (CTR) mode encryption

1. 定義functions: make_iv（合併nonce和counter）、bxor

```python
def make_iv(n, cnt):
    tmp_iv = bytearray(16)
    for i in range(16):
        if i < 8:
            tmp_iv[i] = n[i]
        else:
            tmp_iv[i] = cnt[i-8]
    return tmp_iv

# xor for bytes
def bxor(b1, b2):
    result = b""
    for b1, b2 in zip(b1, b2):
        result += bytes([b1 ^ b2])
    return result
```

2. 設定vector、nonce、counter

```python
# create a 16 byte array to hold IV
initial_vector = bytearray(16)

# Randomly generate a nounce
nounce = b',\xd5{\xe95\x86B\xa1'

# Initialize counter
counter = 0
counter_bytes = counter.to_bytes(8, 'big')

# Initialize nounce
initial_vector = make_iv(nounce, counter_bytes)
```

3. 明文切塊

```python
# divide plaintext into 16-byte blocks
plain_text_blocks = []
with open(tmpPic, 'rb') as f:
    while True:
        buf = f.read(16)
        if buf != b'':
            plain_text_blocks.append(buf)
        if not buf:
            break
```

4. 設定key、AES cipyer、輸出檔

```
key = b'MORECOFFEEPLEASE'
cipher = AES.new(key, AES.MODE_ECB)
output = open("out_ctr_enc.ppm", "wb")
output.close()
output = open("out_ctr_enc.ppm", "ab")
```

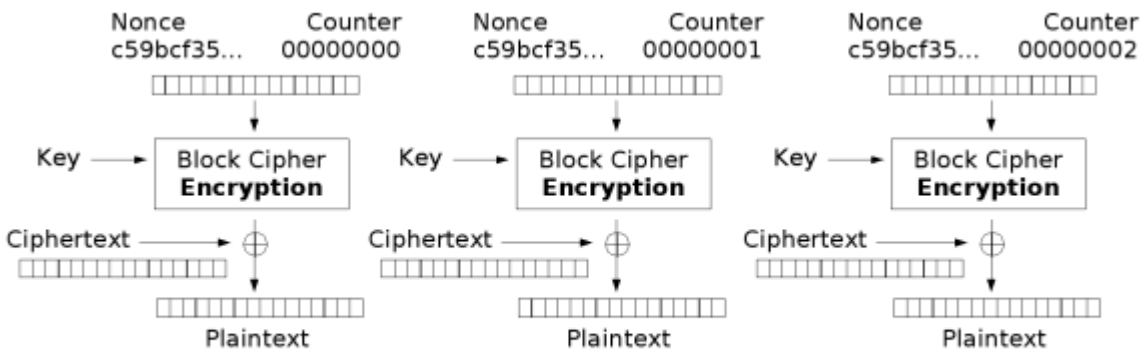
5. 對每塊加密vector後和明文xor，得到密文，寫檔

```
for i in range(len(plain_text_blocks)):
    if i == 0:
        output.write(plain_text_blocks[i])
    else:
        msg = cipher.encrypt(initial_vector)
        msg_bin = BitArray(hex=msg.hex()).bin
        cipher_text_block = bxor(msg, plain_text_blocks[i])
        output.write(cipher_text_block)
    counter += 1
    initial_vector = make_iv(nounce, counter.to_bytes(8, 'big'))
```

6. 將ppm轉成jpeg，存檔

```
# Convert to jpeg
Picture = "./out_ctr_enc.ppm"
im = Image.open(Picture)
im.save("./test_enc/CTR.jpeg")
```

解密



Counter (CTR) mode decryption

1. 設定輸入和輸出檔案、key、AES cipyer、nonce、counter

```python
input = "./out_ctr_enc.ppm"

output = open("out_ctr_dec.ppm", "wb")
output.close()
output = open("out_ctr_dec.ppm", 'ab')
line_count = 0
key = b'MORECOFFEEPLEASE'
nounce = b',\xd5{\xe95\x86B\xa1'
cipher = AES.new(key, AES.MODE_ECB)
```

```python
# Initialize counter
counter = 0
counter_bytes = counter.to_bytes(8, 'big')
```

2. 定義functions: make_iv（合併nonce和counter）、bxor

```python
def make_iv(n, cnt):
    tmp_iv = bytearray(16)
    for i in range(16):
        if i < 8:
            tmp_iv[i] = n[i]
        else:
            tmp_iv[i] = cnt[i-8]
    return tmp_iv

# xor for bytes
def bxor(b1, b2):
    result = b""
    for b1, b2 in zip(b1, b2):
        result += bytes([b1 ^ b2])
    return result
```

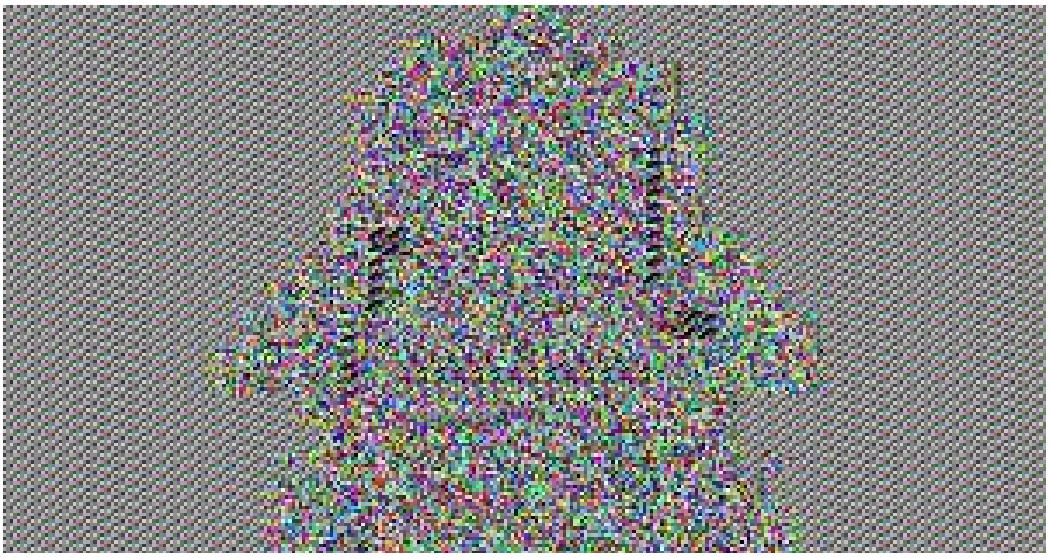
3. 對每塊加密vector後和密文xor，得到明文，寫檔

```
    initial_vector = make_iv(nounce, counter_bytes)
    with open(input, 'rb') as f:
        while True:
            buf = f.read(16)
            if counter == 0:
                output.write(buf)
            else:
                msg = cipher.encrypt(initial_vector)
                decipher_text = bxor(msg, buf)
                output.write(decipher_text)
            counter += 1
            initial_vector = make_iv(nounce, counter.to_bytes(8, 'big'))
            if not buf:
                break
```

4. 將ppm轉成jpeg，存檔

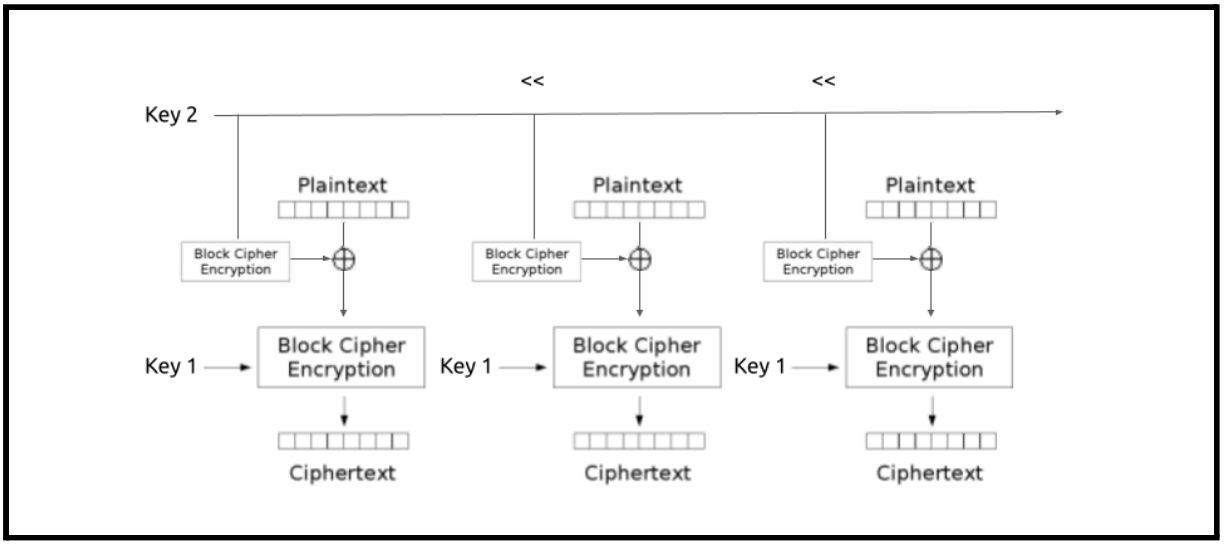
```
    # Convert to jpeg
    Picture = "./out_ctr_dec.ppm"
    im = Image.open(Picture)
    im.save("./test_dec/CTR.jpeg")
```

CUSTOM 模式(自行設計)

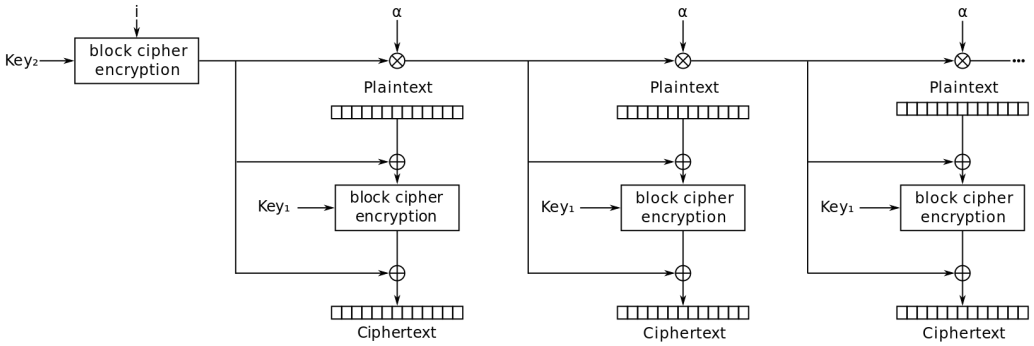


ECB模式下，同樣的明文塊會被加密成相同的密文塊，加密後(上圖)仍然可以看出原圖輪廓，無法提供嚴格的資料保密性，於是我們決定設計新的模式改善這點。

加密



參考XTS模式的作法，如下圖

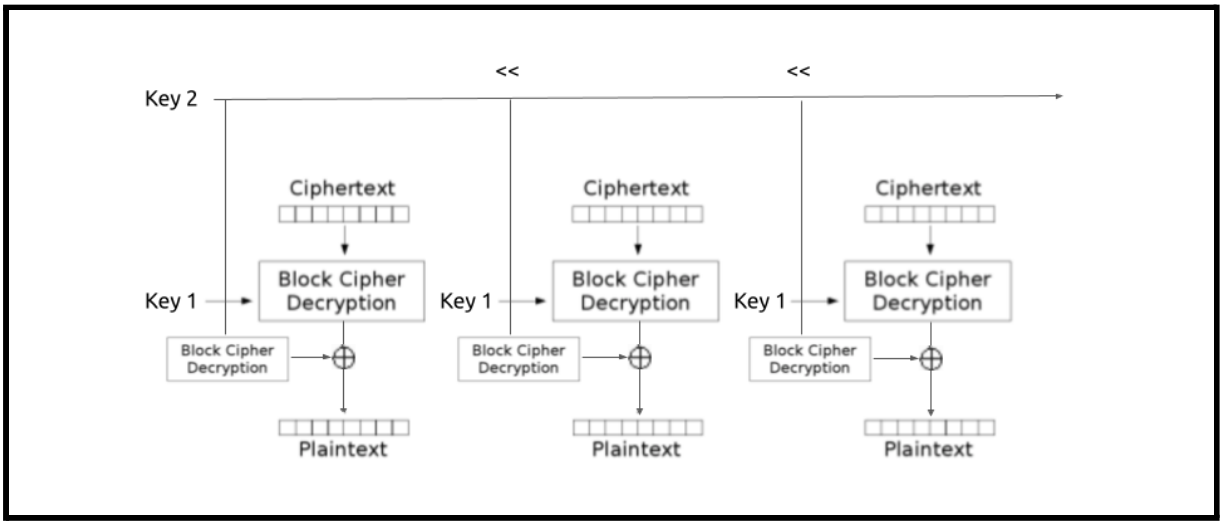


加入key2並持續更新key2(我們採用循環左移的方式)、加密key2，將加密後的key2和每塊明文進行xor，再對處理過後的明文加密，得到具**偽隨機性的密文(見下圖)**。

實作如下:

```python
    with open(tmpPic, 'rb') as f:
        while True:
            buf = f.read(16)
            if line_count == 0:
                msg = buf
            else:
                #對key2加密
                key2 = cipher.encrypt(key2)
                #加密後的key2和明文xor
                buf = byte_xor(buf, key2)
                #對處理後的明文加密
                msg = cipher.encrypt(buf)
                #key2循環左移
                key2 = make_key2(key2)
            output.write(msg)
            line_count += 1
            if not buf:
                break
```

解密



解密過程和加密類似，一樣持續更新key2(循環左移)、加密key2，先將密文解密，再與加密後的key2進行xor，得到明文。

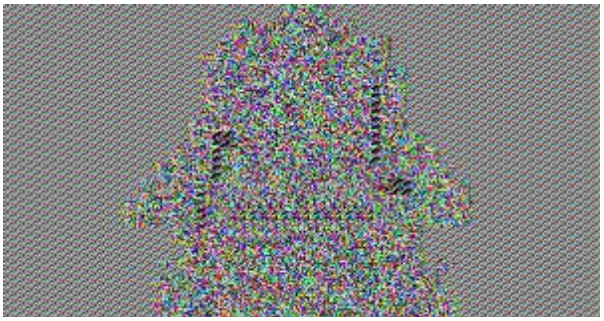
實作如下:

```python
    with open(input, 'rb') as f:
        while True:
            buf = f.read(16)
            if line_count == 0:
                msg = buf
            else:
                #解密key2
                key2 = decipher.encrypt(key2)
                #解密密文
                msg = decipher.decrypt(buf)
                #兩者xor
                msg = byte_xor(msg, key2)
                #key2循環左移
                key2 = make_key2(key2)
            output.write(msg)
            line_count += 1
            if not buf:
                break
```

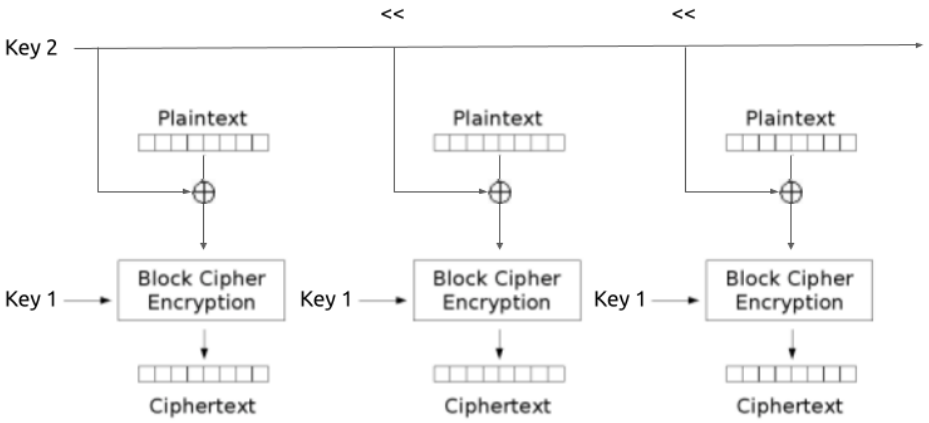
困難與心得

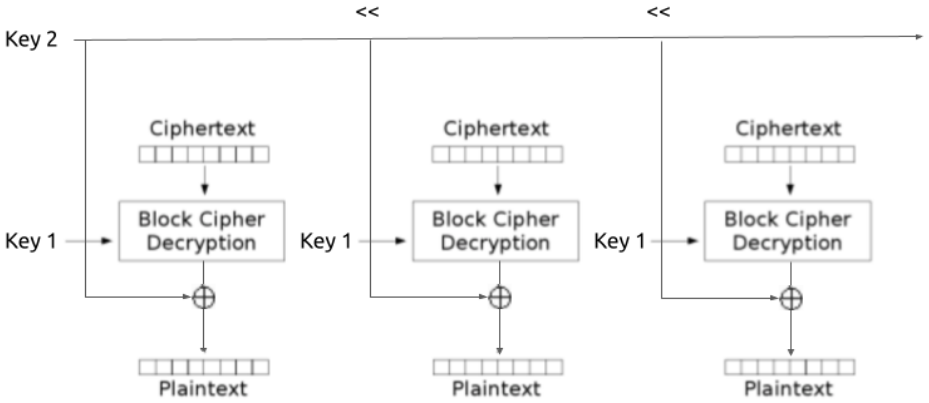
剛開始嘗試改善ECB的的時候加入了key2的概念，使用循環左移更新key2，但結果不盡理想，讓我們有點灰心，持續腦力激盪才得到CUSTOM最終版本。

左為ECB加密結果、右為初版CUSTOM加密結果：




初版架構如下：


<<
<<
Key 2
Plaintext
Plaintext
Plaintext
Key 1
Block Cipher Encryption
Key 1
Block Cipher Encryption
Key 1
Block Cipher Encryption
Ciphertext
Ciphertext
Ciphertext


<<
<<
Key 2
Ciphertext
Ciphertext
Ciphertext
Key 1
Block Cipher Decryption
Key 1
Block Cipher Decryption
Key 1
Block Cipher Decryption
Plaintext
Plaintext
Plaintext