密码学原理
实验二：对称加密与认证

学号： 2022113564　　姓名： 张哲恺

实验目的：本实验旨在掌握运用密码学工具实现 CPA 安全加密与 CCA 安全加密，并采用 CCA 能力攻击 CPA 安全加密方案。

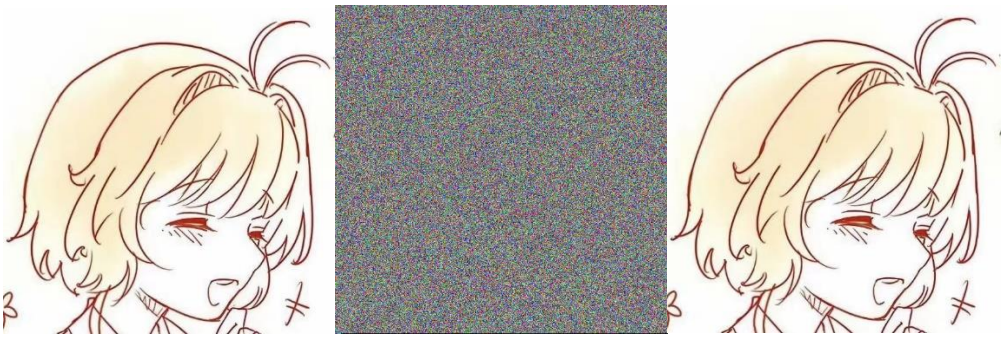
1、 使用密码学工具实现 CPA 安全加密算法

(1) 使用密码学工具实现 CPA 安全加密方案

要求：选择 CPA 安全的加密方案对一个图片内容进行加密和解密，密文文件可用图片浏览器打开。

加密时先使用 get_random_bytes 创建初始向量 iv 和密钥 key，使用 cv 库读取图片并转化为字节流，填充后对图片数据使用 AES_CBC 进行加密，再将密文与初始向量和填充进行拼接，再使用 np.frombuffer 转化回三维数组并存储在新的图像中。

解密时同样先读取图片数据，并分离初始向量 iv 和密文以及填充块，使用相同的密钥对密文进行解密得到明文。



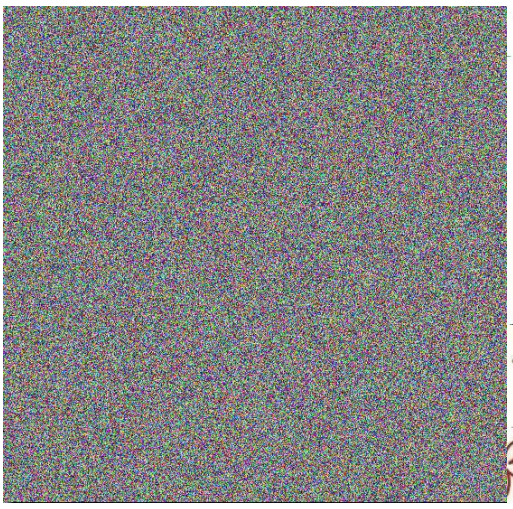
原图　　加密后　　解密后

2、 采用 CCA 攻击分析 CPA 安全加密方案

(1) 利用 CCA 能力敌手攻击 CPA 安全的加密方案

要求：对上一步中 CPA 安全加密方案加密的图片文件进行 CCA 攻击：篡改密文图片，然后用解密预言机对篡改图片解密。

读取密文图片，随机选取 100 个像素点进行篡改，然后使用同样的解密预言机解密得到解密成果，发现篡改成功，CCA 攻击成功。



篡改后的密文文件

解密结果

局部放大

3、 使用密码学工具实现 CCA 安全的加密算法

(1) 使用密码学工具实现 CCA 安全加密方案

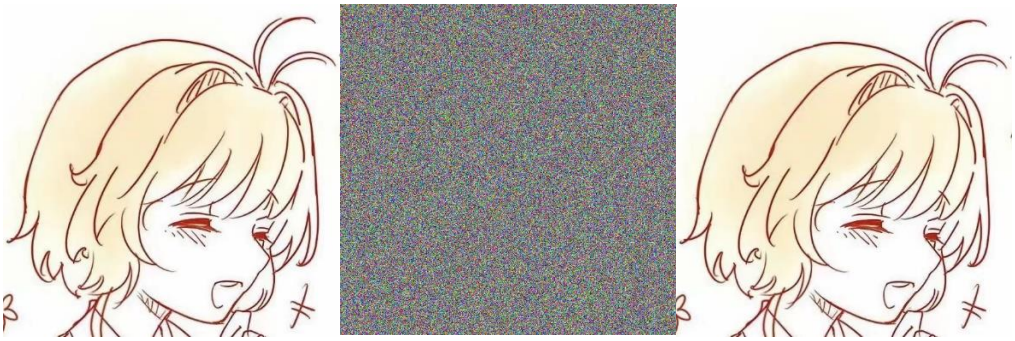
要求：选择 CCA 安全的加密方案对一个图片内容进行加密和解密，密文文件可用图片浏览器打开。

使用 AES_GCM 加密方案，先随机生成长度为 128 的密钥，再读取 bmp 格式图像文件，分离文件头部分和图像数据部分。对其中的图像数据部分进行加密，并对加密结果 hash 得到 hmac 用于解密时的 MAC 验证。然后将 nonce 和 hmac 拼接在文件头之后，接着修改文件头，将图像数据

偏移量修改为 98，再将密文拼接进去，得到的整个图像数据写入密文文件中。

解密时，读取图像数据，分别获取文件头，nonce，hmac 和密文，对 hmac 进行验证，如果密文 hash 结果与 hmac 相等，验证通过即可解密，否则不解密。

```
key:  b'4\xb2\xd4y\xa8{&\xd2\rE~\xf9\xbfV\x1c\xc9'
nonce:  97B63918E487D6452EB3E1D1
hmac_value:  56350B02F9BC44B84344E2AD09873D8F3BE7EE99D2745C6E47FC2B5281E60B72
验证通过，解密成功
```



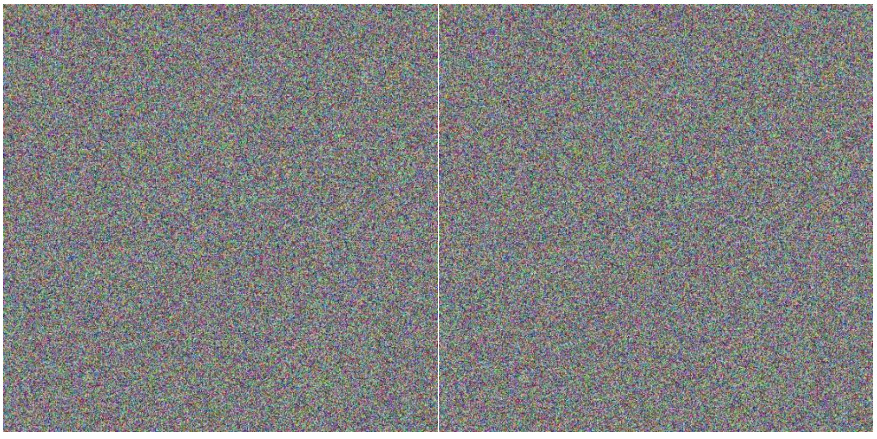
原图　　加密后　　解密后

(2) CCA 敌手能否攻击成功 CCA 安全的加密方案

要求：尝试用第 2 步 CCA 攻击来攻击 CCA 安全加密的图片。

同样先读取密文文件，获取文件头，nonce 和 hmac 部分 以及密文数据，随机选取 100 个像素点进行篡改，并使用解密预言机对篡改结果进行解密。Hmac 验证不通过解密失败，CCA 攻击失败。

```
hmac:  56350B02F9BC44B84344E2AD09873D8F3BE7EE99D2745C6E47FC2B5281E60B72
hmac_value:  A0D996BE2DC2092D54665444F93F906F495061723D858AEA810D621671560A72
验证失败，解密失败
```



篡改后的密文文件　　解密结果

CPA.py:

```
import sys
import cv2
import numpy as np
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes

def encrypt_image(input_image_path, output_image_path, key , iv):

    input_image = cv2.imread(input_image_path)
    input_row , input_column , input_depth = input_image.shape

    input_image_bytes = input_image.tobytes()


    cipher = AES.new(key , AES.MODE_CBC , iv)

    image_bytes_padded = pad(input_image_bytes , 16) #填充
    cihpertext = cipher.encrypt(image_bytes_padded) #加密

    padsize = len(image_bytes_padded) - len(input_image_bytes)

    _pad = input_column * input_depth - 16 - padsize
    ciphertext_padded = iv + cihpertext + bytes(_pad) #拼接

    finally_ciphertext = np.frombuffer(ciphertext_padded , dtype=input_image.dtype).reshape(input_row + 1 , input_column ,
input_depth) #转化

    cv2.imwrite(output_image_path, finally_ciphertext) #保存

def decrypt_image(input_image_path, output_image_path, key):
    encrypt_image = cv2.imread(input_image_path)
    encrypt_image_bytes = encrypt_image.tobytes()

    encry_row , encry_column , encry_depth = encrypt_image.shape

    iv = encrypt_image_bytes[:16]

    input_image_size = ((encry_row - 1) * encry_column * encry_depth)
    padsize = (input_image_size // 16 + 1) * 16 - input_image_size

    ciphertext = encrypt_image_bytes[16 : 16 + input_image_size + padsize]

```

```
    cipher = AES.new(key , AES.MODE_CBC , iv)
    plaintext = cipher.decrypt(ciphertext)

    plaintext_unpad = unpad(plaintext , 16)

      plaintext_image  =  np.frombuffer(plaintext_unpad  ,  encrypt_image.dtype).reshape(encry_row  -  1  ,  encry_column  ,
encry_depth)

    cv2.imwrite(output_image_path, plaintext_image)


key = get_random_bytes(16)
iv = get_random_bytes(16)
print(key)
# 加密图像
                                          encrypt_image('C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\input_image.jpeg',
'C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\encrypted_image.bmp', key , iv)

# 解密图像
                                       decrypt_image('C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\encrypted_image.bmp',
'C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\decrypted_image.bmp', key)
#key: b'\xda\xf5\nG\xe0\xf7\tN\xfa \x10\xfe\xaa\xb4\xc6\xe1'

```

CCA_Attack1.py:

```
import cv2
import numpy as np
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
from Crypto.Random import get_random_bytes
import random

def decrypt_image(input_image_path, output_image_path, key):
    encrypt_image = cv2.imread(input_image_path)

    encrypted_image = cv2.imread(input_image_path)
    if encrypted_image is None:
        print("Error: Unable to read the input_image")
    else:
        print("Image loaded successfully")

    encrypt_image_bytes = encrypt_image.tobytes()

    encry_row, encry_column, encry_depth = encrypt_image.shape

```

```
    iv = encrypt_image_bytes[:16]

    input_image_size = ((encry_row - 1) * encry_column * encry_depth)
    padsize = (input_image_size // 16 + 1) * 16 - input_image_size

    ciphertext = encrypt_image_bytes[16 : 16 + input_image_size + padsize]

    cipher = AES.new(key, AES.MODE_CBC, iv)
    plaintext = cipher.decrypt(ciphertext)

    plaintext_unpad = unpad(plaintext, 16)

    plaintext_image = np.frombuffer(plaintext_unpad, encrypt_image.dtype).reshape(encry_row - 1, encry_column, encry_depth)

    cv2.imwrite(output_image_path, plaintext_image)

def perform_CCA_attack(input_image_path, output_image_path, key):
    # 加载密文图片
    encrypted_image = cv2.imread(input_image_path)

    # 确定篡改范围，排除初始向量部分
    height, width, _ = encrypted_image.shape
    tamper_range = [(i, j) for i in range(height) for j in range(width) if not (i == 0 and j < 16)]

    # 随机选择多个像素点进行篡改
    num_pixels_to_tamper = min(len(tamper_range), 100)  # 选择最多 100 个像素点进行篡改
    pixels_to_tamper = random.sample(tamper_range, num_pixels_to_tamper)
    for pixel in pixels_to_tamper:
        encrypted_image[pixel[0], pixel[1], random.randint(0, 2)] = random.randint(0, 255)

    print("Tampering successful")

    # 保存篡改后的图片
    cv2.imwrite(output_image_path, encrypted_image)

    # 使用解密函数对篡改图片进行解密
    decrypt_image(output_image_path, 'C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\decrypted_image_after_attack.bmp', key)

key = b'q\xe0gY\x98\xc0\xc6\xbf\x18\xfa\x1au/\xbb|\xe8'

# 进行 CCA 攻击并尝试解密
                                    perform_CCA_attack('C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\encrypted_image.bmp',
'C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\tampered_encrypted_image.bmp', key)

```

CCA.py:

```python
from cryptography.hazmat.primitives.ciphers.aead import AESGCM
import numpy as np
from PIL import Image
import os
import hmac

def encrypt(key) :
    with open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\CCA\\input_image.bmp" , 'rb') as f:
        input_image = bytearray(f.read())

    input_image[10] = 98 #修改图片数据起始位置

    plaintext = np.array(input_image[55: ]) #记录图片数据
    #加密过程
    aesgcm = AESGCM(key)
    print("key: " ,  key)
    nonce = os.urandom(12)
    print("nonce: " , nonce.hex().upper())
    ciphertext = aesgcm.encrypt(nonce , plaintext.tobytes() , None)
    hmac_value = hmac.new(key, ciphertext, digestmod='sha256').digest()
    print("hmac_value: " , hmac_value.hex().upper())

    with open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\CCA\\encrypted_image.bmp" , "wb") as f :
        f.write(input_image[:55]) #文件头
        f.write(nonce) #nonce
        f.write(hmac_value) #HMAC 消息
        f.write(ciphertext) #加密图像数据

def decrypt(key) :
    aesgcm = AESGCM(key)
    with open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\CCA\\encrypted_image.bmp" , "rb") as f :
        bmphead = bytearray(f.read(55)) #获取文件头
        nonce = bytearray(f.read(12)) #nonce
        # print(nonce.hex().upper())
        hmac_v = bytearray(f.read(32)) #hmac
        # print(hmac_v.hex().upper())
        encrypted_image = bytearray(f.read())
    hmac_value = hmac.new(key , encrypted_image , digestmod='sha256').digest()
    if (hmac_v == hmac_value) :
        bmphead[10] = 54 #修改图像数据起始位置
        ciphertext = np.array(encrypted_image)
        decrypted_img = aesgcm.decrypt(nonce , ciphertext.tobytes(), None) #解密
        with open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\CCA\\decrypted_image.bmp" , "wb") as f :
```

```
            f.write(bmphead)
            f.write(decrypted_img)
        print("验证通过，解密成功")
    else :
        with open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\CCA\\decrypted_image.bmp" , "wb") as f :
            f.write(bmphead)
            f.write(nonce)
            f.write(hmac_v)
            f.write(encrypted_image)
        print("验证失败，解密失败")

key = AESGCM.generate_key(bit_length=128)
encrypt(key)
decrypt(key)

```

CCA_Attack2.py:

```
import random
from cryptography.hazmat.primitives.ciphers.aead import AESGCM
import numpy as np
from PIL import Image
import os
import hmac

def modify_image(width , height , image_data, num_pixels):
    total_pixels = height * width

    pixel_indices = random.sample(range(total_pixels), num_pixels)
    for index in pixel_indices:
        row = index // width
        col = index % width
        pixel_index = (row * width + col) * 3

        red = random.randint(0, 255)
        green = random.randint(0, 255)
        blue = random.randint(0, 255)

        image_data[pixel_index:pixel_index + 3] = bytes([red, green, blue])

        return image_data

def decrypt(key , input_file , output_file) :
    aesgcm = AESGCM(key)
    with open(input_file , "rb") as f :
        bmphead = bytearray(f.read(55)) #获取文件头
```

```
        nonce = bytearray(f.read(12)) #nonce
        # print(nonce.hex().upper())
        hmac_v = bytearray(f.read(32)) #hmac
        print("hmac: " , hmac_v.hex().upper())
        encrypted_image = bytearray(f.read())
    hmac_value = hmac.new(key , encrypted_image , digestmod='sha256').digest()
    print("hmac_value: " , hmac_value.hex().upper())
    if (hmac_v == hmac_value) :
        bmphead[10] = 54 #修改图像数据起始位置
        ciphertext = np.array(encrypted_image)
        decrypted_img = aesgcm.decrypt(nonce , ciphertext.tobytes(), None) #解密
        with open(output_file , "wb") as f :
            f.write(bmphead)
            f.write(decrypted_img)
        print("验证通过，解密成功")
    else :
        with open(output_file , "wb") as f :
            f.write(bmphead)
            f.write(nonce)
            f.write(hmac_v)
            f.write(encrypted_image)
        print("验证失败，解密失败")

with open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\CCA\\encrypted_image.bmp", "rb") as f:
    encrypted_image_data = bytearray(f.read())

bmphead = encrypted_image_data[:98]
image_data = encrypted_image_data[98:]

image_data = modify_image(int.from_bytes(bmphead[18:22], byteorder='little') , int.from_bytes(bmphead[22:26],
byteorder='little') ,image_data, 100)

tampered_encrypted_image_data = bmphead + image_data
tampered_file = "C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\CCA\\tampered_encrypted_image.bmp"
with open(tampered_file, "wb") as f:
    f.write(tampered_encrypted_image_data)
tempered_output_file = "C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\CCA\\tampered_decrypted_image.bmp"

key = b'4\xb2\xd4y\xa8{&\xd2\rE~\xf9\xbfV\x1c\xc9'
decrypt(key , tampered_file , tempered_output_file)
```