《信息安全基础》实验报告

一、实验目的

1. 学习并掌握图像信息隐藏的基本原理和方法

2. 实现基于 LSB 的信息隐藏和提取算法

二、实验项目内容

1. 使用 LSB 算法在图片中隐藏如下信息：CQUWATERMASKEXP；

2. 从被隐藏数据的图片中解析出如上信息；

3. 在实际应用中，隐藏信息量通常是不可预知的，同时，攻击者也很容易从最低位像素提取到隐藏信息并对此进行密文分析。另一方面，如何确保信息来源于正确的发送者？针对这些问题，请设计完整的方案。

三、实验设计

1. LSB 算法实验原理

任何多媒体信息在数字化时都会产生物理随机噪声，而人的感官系统对这些随机噪声并不敏感，通过使用秘密信息比特替换随机噪声，从而实现信息隐藏。在图像中，高位平面对图像感官质量起主要作用，去除图像最低几个位平面并不会造成画面质量的下降。利用这个原理可用秘密信息（或称水印信息）替代载体图像低位平面以实现信息嵌入。

LSB 算法选用最低位平面来嵌入信息，最低位平面对图像的视觉效果影响最轻微，因此在视觉上很难察觉。作为大数据量的信息隐藏方法，LSB 在保密通信中仍占据相当重要的地位。

LSB 算法加密和解密步骤如下：

1.1 LSB 加密：

（1）读入图片；

（2）准备待隐藏的信息，将其转换为二进制；

（3）遍历图像，对像素的最低 1bit 置 0，同时在该比特位写入 1 位二进制表示隐藏的信息。

1.2 LSB 解密：

（1）预知隐藏信息量（等同于 key）；

（2）提取出像素的最低 1bit，组合成连续 bit 数据，转换为 ASCII 码对比

是否与隐藏信息一致。

2. 基于加密和身份验证的信息隐藏

在实际应用中，想要实现信息隐藏量的传递其实很容易，可以约定在前 256 个像素中隐藏图片携带的隐藏信息量。至于保护隐藏的信息防止泄露，可以在隐藏前进行加密，本实验采用的是 RSA 非对称加密。要想确保信息来源于正确的发送者，可以在加密前加入身份认证，如 RSA 非对称加密。综上所述，对于要隐藏的信息 “CQUWATERMASKEXP”，可以先使用发送方的私钥进行加密，然后使用接收方的公钥进行加密，再隐藏进图片中，再由接收方先使用私钥解密以得到明文，后使用发送方的公钥解密认证身份，解密完成后即可得到隐藏的信息。详见代码实现。

四、实验过程或算法

1. LSB 算法加密解密

```
from PIL import Image


def hide_message(image_path, message):
    """LSB 在图像中隐藏信息
        args:
            image_path: 图像文件路径
            message: 待隐藏的信息
        return:
            hidden_image: 隐藏信息后的图像对象
    """
    # 打开图像
    image = Image.open(image_path)
    binary_message = ''.join(format(ord(c), '08b') for c in
message)
    print("原始信息: ", binary_message)
    hidden_image = image.copy()
    # 确定大小
    width, height = image.size
    total_pixels = width * height
```

```
    key = len(binary_message)   # 用于确定隐藏信息量的密钥
    if key > total_pixels:
        raise ValueError("Key is larger than total pixels in the
image.")
    # 信息隐藏
    pixels_processed = 0
    binary_message2 = ""
    for x in range(width):
        for y in range(height):
            if pixels_processed < key:
                pixel = list(image.getpixel((x, y)))    # RGB 的 3
个 8bit 像素
                for i in range(len(pixel)):
                    pixel[i] &= 0xFE                    # 将像素的
最低 bit 置 0
                    pixel[i] |=
int(binary_message[pixels_processed], 2)    # 将隐藏信息写入像素的最
低 bit
                    pixels_processed += 1
                    binary_message2+=bin(pixel[i])[-1]
                hidden_image.putpixel((x, y), tuple(pixel)) # 将
(x,y)位置的像素值更新:[r,g,b]=tuple(pixel)
    print("隐藏信息: ", binary_message2)
    return hidden_image


def extract_message(hidden_image_path, key):
    """LSB 从图像中提取隐藏信息
        args:
            hidden_image_path: 隐藏信息后的图像文件路径
            key: 用于确定隐藏信息量的密钥
        return:
            extracted_message: 提取出的信息
```

```
    """
    # 打开图像
    hidden_image = Image.open(hidden_image_path)
    image = hidden_image.copy()
    # 确定大小
    width, height = image.size
    total_pixels = width * height
    if key > total_pixels:
        raise ValueError("Key is larger than total pixels in the image.")
    # 提取信息
    extracted_message = ""
    pixels_processed = 0
    binary_message = ""
    for x in range(width):
        for y in range(height):
            if pixels_processed < key:  # 否则读取字符串过长
                pixel = list(image.getpixel((x, y)))
                for i in range(len(pixel)):
                    extracted_bit = pixel[i] & 0x01          # 提取像素的最低 1bit
                    binary_message += str(extracted_bit)    # 将提取的 bit 添加到二进制消息中
                    pixels_processed += 1
    print("提取信息: ", binary_message)
    # 将二进制消息转换为 ASCII 码
    for i in range(0, len(binary_message), 8):
        char = chr(int(binary_message[i:i+8], 2))
        extracted_message += char
    return extracted_message
```

```python
if __name__ == "__main__":
    # 读入图像
    image_path = r"pic\train.png"
    # 待隐藏的信息和密钥
    message = "CQUWATERMASKEXP"

    # 调用 hide_message() 函数进行信息隐藏
    hidden_image = hide_message(image_path, message)
    # 保存隐藏信息后的图像
    hidden_image_path = r"pic\hidden_train.png"
    hidden_image.save(hidden_image_path)

    # 调用 extract_message() 函数进行信息提取
    key = len(''.join(format(ord(c), '08b') for c in message))
    extracted_message = extract_message(hidden_image_path, key)

    print("提取字符串: ", extracted_message)
```

以 CQUWATERMASKEXP 为例，输出结果如下：

```
原始信息:  0100001101010001010101010101011101000001010101000100010101001001001101010000010101001101001011010001010101100001010000
隐藏信息:  0100001101010001010101010101011101000001010101000100010101001001001101010000010101001101001011010001010101100001010000
提取信息:  0100001101010001010101010101011101000001010101000100010101001001001101010000010101001101001011010001010101100001010000
提取字符串:  CQUWATERMASKEXP
```

可以看到，隐藏信息与提取信息相同，加密解密后消息信息不变。

隐藏前后的图片对比如下，可以看到，LSB 对图片几乎没有产生视觉上的影响：

2. 基于加密和身份验证的信息隐藏

```
from PIL import Image
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_v1_5


# 加密函数
def encrypt(message, public_key):
    cipher = PKCS1_v1_5.new(public_key)      # 不限制明文长度
    ciphertext = cipher.encrypt(message)
    return ciphertext
```

```
# 解密函数
def decrypt(ciphertext, private_key):
    cipher = PKCS1_v1_5.new(private_key)
    message = cipher.decrypt(ciphertext, None)
    return message


def hide_message(image_path, message):
    """LSB 在图像中隐藏信息
        args:
            image_path: 图像文件路径
            message: 待隐藏的加密过的信息(二进制串形式)
        return:
            hidden_image: 隐藏信息后的图像对象
```

```
    """

    # 打开图像

    image = Image.open(image_path)

    print("原始信息: ", message)

    hidden_image = image.copy()

    # 确定大小

    width, height = image.size

    total_pixels = width * height

    key = len(message)   # 用于确定隐藏信息量的密钥

    if key > total_pixels:

        raise ValueError("Key is larger than total pixels in the
image.")

    # 信息隐藏

    pixels_processed = 0

    message2 = ""

    for x in range(width):

        for y in range(height):

            pixel = list(image.getpixel((x, y)))    # RGB 的 3 个
8bit 像素

            for i in range(len(pixel)):

                if pixels_processed < key:

                    pixel[i] &= 0xFE                    # 将像素的
最低 bit 置 0

                    pixel[i] |= int(message[pixels_processed],
2)    # 将隐藏信息写入像素的最低 bit

                    pixels_processed += 1

                    message2+=bin(pixel[i])[-1]

            hidden_image.putpixel((x, y), tuple(pixel)) # 将(x,y)
位置的像素值更新:[r,g,b]=tuple(pixel)

    print("隐藏信息: ", message2)

    return hidden_image
```

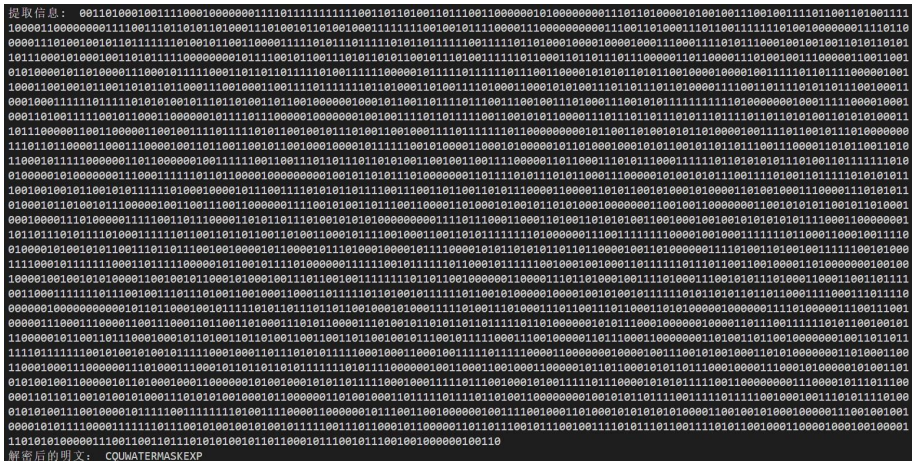
```python
def extract_message(hidden_image_path, key):
    """LSB 从图像中提取隐藏信息
        args:
            hidden_image_path: 隐藏信息后的图像文件路径
            key: 用于确定隐藏信息量的密钥
        return:
            extracted_message: 提取出的信息
    """
    # 打开图像
    hidden_image = Image.open(hidden_image_path)
    image = hidden_image.copy()
    # 确定大小
    width, height = image.size
    total_pixels = width * height
    if key > total_pixels:
        raise ValueError("Key is larger than total pixels in the image.")
    # 提取信息
    pixels_processed = 0
    binary_message = ""
    for x in range(width):
        for y in range(height):
            pixel = list(image.getpixel((x, y)))
            for i in range(len(pixel)):
                if pixels_processed < key:  # 否则读取字符串过长
                    extracted_bit = pixel[i] & 0x01          # 提取像素的最低 1bit
                    binary_message += str(extracted_bit)     # 将提取的 bit 添加到二进制消息中
                    pixels_processed += 1
    print("提取信息: ", binary_message)
```

```
    return binary_message


if __name__ == "__main__":
    # 生成 RSA 密钥对
    key1 = RSA.generate(1024)   # A 的密钥用于认证 A
    key2 = RSA.generate(4096)   # B 的密钥用于认证 B
    # 获取公钥和私钥
    public_key1 = key1.publickey()
    private_key1 = key1
    public_key2 = key2.publickey()
    private_key2 = key2
    # 要加密的明文
    plaintext = b"CQUWATERMASKEXP"
    print("明文：", plaintext)


    # 使用 A 的公钥加密明文
    ciphertext1 = encrypt(plaintext, public_key1)
    # 使用 B 的公钥加密明文,结果形式为字节流
    ciphertext2 = encrypt(ciphertext1, public_key2)
    # ciphertext3 = ciphertext2.decode('utf-8')   # 转换为字符串(无法解析)
    binary_ciphertext2 = ''.join(format(b, '08b') for b in ciphertext2)   # 转换为二进制串
    print("密文：", ciphertext2)


    # 读入图像
    image_path = r"pic\train.png"


    # 调用 hide_message() 函数进行信息隐藏
    hidden_image = hide_message(image_path, binary_ciphertext2)
    # 保存隐藏信息后的图像
```

```
    hidden_image_path = r"pic\hidden_train2.png"

    hidden_image.save(hidden_image_path)


    # 调用 extract_message() 函数进行信息提取

    key = len(binary_ciphertext2)

    binary_message = extract_message(hidden_image_path, key)

    integer = int(binary_message, 2)

    extracted_message = integer.to_bytes((integer.bit_length() +
7) // 8, byteorder='big')


    # 使用 B 的私钥解密密文

    decrypted_text2 = decrypt(extracted_message, private_key2)

    # 使用 A 的私钥解密密文

    decrypted_text1 = decrypt(decrypted_text2, private_key1)


    print("解密后的明文：", decrypted_text1.decode('utf-8'))
```

以 CQUWATERMASKEXP 为例，输出结果如下：

```
明文：  b'CQUWATERMASKEXP'
密文：  b'4O\x10\x1e\xff\xe6\xd3s\x02\x80v\x85\'\'\xb3O\x86\x01\xe7kGKH\xff%\xe1\xc0\x0eh\xec\xfe\x90\x1e\xc1\xd2[\xfaY\x87\xd7}o\xcf\xb4B\x
11\xc7\xae$\x9a\xd6\xe2\x89\xaf\x80^Y\xd6\xb2\xe9\xfb\x1bw\x06\xc3\xa4\xe0\xcc\xa8Z\x1c_\x1bo\xa7\xe0\xbe\xfd\xcc*\xd6B\x13\xed\xe0\x98\xc9
f\xb69\x19\xef\xed\x1az1S\xb7hy\xbdnF#\xf7\xd5.\xd3d\x08\xb3{\x9c\x9d\x1c\xaf\xfe\x80\x8f\x84F\x9f,`^\xe0\x80\x93\xdb\xe6V\x1d\xdb\xae\xf6\
xd4\xd5\x1d\xc1\x982{\xeb%\xd3#\xdf\xd8\x02\xcd+B{.\x80\xed\x868M\x99dB\xfc\xa1\x8a\x0bDV[s\x86\xb3X\xbe\x06\xc0\x9f\x99\xdb\xb52g\x83c\xae
?j\xbao\xe9\x05\x01\xc7\xed\x84\x01-t\x06\xf5\xd68)\\\xf4\xdfU\xc9,\xaf\xd1\x0b\x9e\xad\xe76k\x86\x1a\xca(i\x1c:\xb4Z\\\x139\x81\xe570\xd1K
Q\x01\x93\x01\x95e\xa2!\xd0|\xdc5\xbaU\x00{\x8ci\xaadIUx\xc0m\xd7\xa3\xf6m\x9ab\xf23_\xe8\x1c\xff\t\x1f\xd8\xc4\xf2\x14\xac\xed\xc9\x0b\x0b
\xa2\x17\x85j\xdb\t\xa0zi?(\xf1\x7f\x1b\xe0\xb2\xf4\x0f\xcb\xf6/\xc8\x91\xbfvd4\x04\x90\x92\xa1\x92\xc5\x13\xb2\x7f\xb6@\xc3\xb4OG+\xa3\x19
\xbc\xc7\xf7\'t\xc8\xc6\xfbK\xf6PBR\xfdk1x\xef\x01\x00\x16\xc4\xbe\xb71\x8a>\x9d\x1d\x9d\x8dA\x03\xd0s\x90q\xc38\xd9\xa3\xac:Z\xdb\xed\x02\
xb8\x81\r\xcf\xd6K\x82\xcd\xc4Zm36K\x97\xc7 \xdc`4\xd9\x016\xfd\xfc\xa5/\x88\xdd_\x11\x89\xf7\xc3\x01\t\xcaF\xa04f#\x81\xd1\xc5\xb6\xbf\xaf
\x02c#\x05\xb1[\x88qAMR`\xb4F\x05"\xb7\xc4}\xc8\xa7\xdc*\xf9\x80p\xbb\x83l\xa5\x1dR,\r#}\xed0\t[\xcf\xbeD\xeb\xd1S\x90\xbe\x7f\xa7\x86\x05\
xcc\x81<\x8d\x15T2Q\x07$\x85x\x7fr\x92\x97\xce\xc5\x83n\\\x9e\xbb=da\x12\x1dPs7T\xb6.\\\x90&'
原始信息：  001010001001111000010000000111101111111111001101101001101110011000000101000000011101101000101001001110010011101101101001111
100001100000000011110011101101011010001111010010111010010011110000111000000000111001101000111011001111111101001000000011110110
000011101001001011011111110100101100110000111110101110111101101101111110011111011010000100001000011000111101011100010010010011010110101
101110001010001001101011111000000001011110010111010110101100101110100111111011000111011011101110000011011000011101001001110000011001101
010100001011010000111000101111100011011011011111010011111100000101111011111011111010011000010100110101100100000100010011111011011110000001001
100011001001011001101101101110001110010001100111101111111101101000110100011110000110001011010001111011011101010011100011011110100001100011
000100011111101111101010100101110110100110110010000010001011001101111011110011100100111010001110001010111111111101000000010001111100001000
000110100111110010110001100000010111101110000010000000100100111101111011110011001010110000111011101101110101111011011010010110101000011
101110000011001100000110010011110111111010110010010111010011100011010001001010110100010000000011001100000001100101010100101110100000000
111011011000011000111000010011011001100110010001000101111110010100001100010100000101101000100010110100101101101110011100001010110011010
110001011111000000011011000000100111111001100111011011010010011001111000001101100111000011010101010110011011111111010101010110001111111
010000010100000001110001111101110000100000000010010110101111010000001101111010111010110000111000001010010101110011110010110111110101011
100100100101100101011111101000100001011100111101010111011100111010110111000011000011010110010001010000111010010110100010000111000101011
010001011010010111000001001100111001100000011110010100110111001100001010010010110101000100000011001001100000011001010101100101101000100
000100001110100000011111001101110000110101101110100101010000000001111011000100010110101010101100100010010101010111000011000000001
101101110110111010001111110110011011011001101000110001011110010001100110101111111010000011100111111110001001000111111011000110001011110
010000101001010110011101101110010010000101100000101110100010000101111000010101101011011000010011010000011110100101101111100101000
111100010111111100011011111000001011001011110100000011111100101111101100010111110010001001001101111110111011001000011010000000100100
100001001001010100001100100101100010100001111011001100111111111011010000000110001111011010001001111101000111001010110100011001101111
001100011111111011100100111011010011001000110001101111101010010111111011001010000010001001010011111101011010110110100011100011101110
000000100000000000010110110001001011101101110111010010001000111101001110100011110111011010000101000001110011101000101000000111011000001
000001110001110000011001110001101100110100011101011000011101001101010110111111011010000001010111000100000010000110111001111110101100100101
110000001011001111000100001011010011101100100101000111101001000010111100111010000111001110000011010000111010010111000011000000010011011011
111101111111001010010010011111000100011011010101111000100011100010011111011111000010000010000100111001010010001101010000001101000110
```



可以看到，隐藏信息与提取信息相同，中间过程加密后无法破解，加密解密后消息信息不变。

隐藏前后的图片对比如下，LSB 对图片几乎没有产生视觉上的影响：

五、实验过程中遇到的问题及解决情况

1. 问题：LSB 一开始没有保证提取信息长度和隐藏的信息长度相同，导致读取的字符串过长；

解决办法：只提取相同长度的 01 串进行解密：

```
# 提取信息
extracted_message = ""
pixels_processed = 0
binary_message = ""
for x in range(width):
    for y in range(height):
        pixel = list(image.getpixel((x, y)))
        for i in range(len(pixel)):
            if pixels_processed < key:  # 否则读取字符串过长
                extracted_bit = pixel[i] & 0x01          # 提取像素的最低1bit
                binary_message += str(extracted_bit)     # 将提取的bit添加到二进制消息中
            pixels_processed += 1
```

2. 问题：LSB 一开始使用的是 jpg 图像，但在信息提取的时候无法提取出正确的结果；

```
0100001101010001010101010101011101000001010101000100010101010010010011010100000101010011010001010101100001010000
0100001101010001010101010101011101000001010101000100010101010010010011010100000101010011010001010101100001010000
0110111011011100011011100011011100100100011011011100100100100100100100100011100100
1001000111000011001000111000011011100
提取的信息：  m¸ÜnHÛI$É28Ü
```

解决办法：jpg 图像被计算机默认压缩了，因此损失了大量隐藏信息，将其改为 png 格式即可。

3. 问题：身份验证时使用的是私钥加密，调用 cipher.decrypt 函数解密时出现错误，判断解密密钥不是私钥：

```
Traceback (most recent call last):
  File "d:\PyWorkspace\test2.py", line 42, in <module>
    decrypted_text1 = decrypt(decrypted_text2, public_key1)
                      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "d:\PyWorkspace\test2.py", line 17, in decrypt
    message = cipher.decrypt(ciphertext, None)
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\KSoftware\Python\Lib\site-packages\Crypto\Cipher\PKCS1_v1_5.py", line 180
, in decrypt
    m_int = self._key._decrypt(ct_int)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\KSoftware\Python\Lib\site-packages\Crypto\PublicKey\RSA.py", line 187, in
 _decrypt
    raise TypeError("This is not a private key")
TypeError: This is not a private key
```

解决办法：由于 RSA 中公钥与私钥是相对的，因此将其互换即可：

4. 问题：由于需要调用两次 cipher.encrypt(message)进行加密，第一次加密后的密文过长，无法再次加密：

```
Traceback (most recent call last):
  File "d:\PyWorkspace\test2.py", line 34, in <module>
    ciphertext2 = encrypt(ciphertext1, public_key2)
                  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "d:\PyWorkspace\test2.py", line 11, in encrypt
    ciphertext = cipher.encrypt(message)
                 ^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\KSoftware\Python\Lib\site-packages\Crypto\Cipher\PKCS1_v1_5.py", line 107
, in encrypt
    raise ValueError("Plaintext is too long.")
ValueError: Plaintext is too long.
```

解决办法：在 RSA 加密中，加密的明文长度受到密钥的长度限制，一般情况下，RSA 2048 位密钥对最多支持加密 245 字节的明文。想要长度不受限制，将 cipher = PKCS1_OAEP.new(private_key)改成
cipher = PKCS1_v1_5.new(public_key)即可；

```
# 加密函数
def encrypt(message, public_key):
    cipher = PKCS1_v1_5.new(public_key)      # 不限制明文长度
    ciphertext = cipher.encrypt(message)
    return ciphertext
```

5. 问题：解决问题 4 后还出现过明文过长的问题；

解决办法：将第二个密钥加长即可：

```
# 生成RSA密钥对
key1 = RSA.generate(1024)    # A的密钥用于认证A
key2 = RSA.generate(4096)    # B的密钥用于认证B
```

6. 问题：实现基于身份认证和 RSA 加密的 LSB 时，由于所调用函数返回类型的限制，无法重用原本定义的 LSB 加密、解密函数：

解决办法：输入前就将字节流转换成二进制字符串（不转成字符串是因为没有编码可以将密文解析为字符串），输出后先将二进制串转换为字节流，再解密出明文；

六、实验结果及分析和（或）源程序调试过程

1. 使用 LSB 隐藏和提取结果如下：

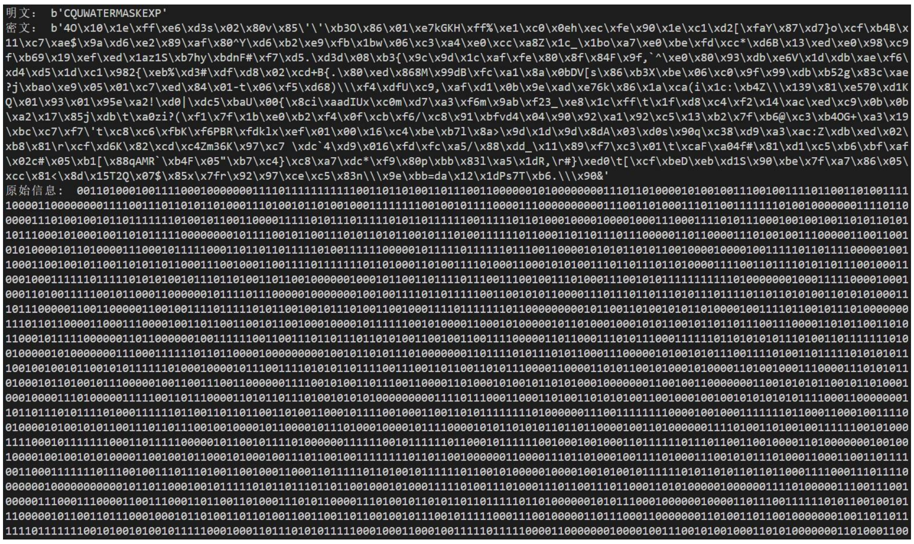
```
原始信息：  0100001101010001010101010101011101000001010101000100010101001001001101010000010101001101001011010001010101100001010000
隐藏信息：  0100001101010001010101010101011101000001010101000100010101001001001101010000010101001101001011010001010101100001010000
提取信息：  0100001101010001010101010101011101000001010101000100010101001001001101010000010101001101001011010001010101100001010000
提取字符串：  CQUWATERMASKEXP
```

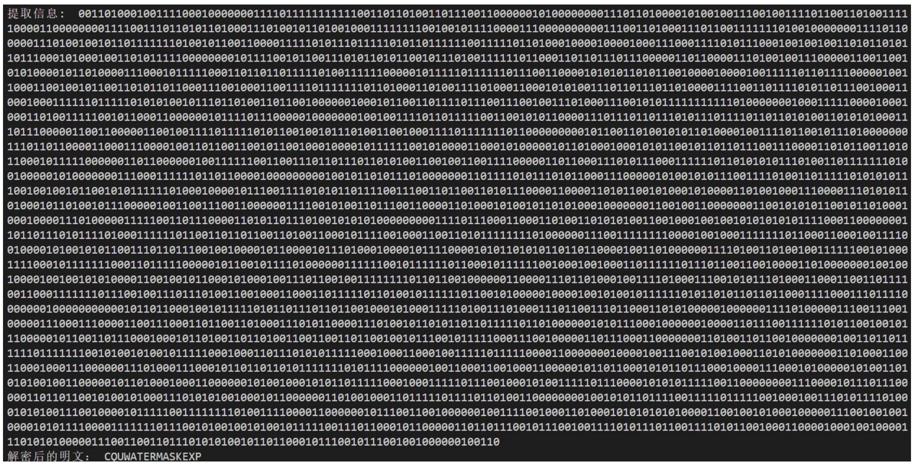
可以看到，隐藏信息与提取信息相同，加密解密后消息信息不变。

隐藏前后的图片对比如下，可以看到，LSB 对图片几乎没有产生视觉上的影响：

2. 基于身份认证和 RSA 加密的 LSB 结果如下：





可以看到，隐藏信息与提取信息相同，中间过程加密后无法破解，加密解密后消息信息不变。

隐藏前后的图片对比如下，LSB 对图片几乎没有产生视觉上的影响：

总的来说，LSB 算法是一种简单的信息隐藏方法，用于在图像、音频、视频等媒体文件中隐藏秘密信息。它的基本思想是将秘密信息的二进制比特位嵌入到媒体文件中的像素、采样点或帧中的最低有效位中，从而不显著改变媒体文件的外观或声音，以达到隐蔽传输秘密信息的目的。

但是，LSB 算法较为简单，对抵抗专业的信息隐藏检测工具可能不够安全。在实际应用中，需要根据具体的信息隐藏威胁和安全需求选择合适的信息隐藏方法，并结合其他信息隐藏技术来提高安全性。为了进行身份认证和信息加密，我还设计了基于身份认证和 RSA 加密的 LSB 算法，并成功实践。