南昌大学实验报告

学生姓名：丁俊　　学　　号：8003119100　专业班级：信息安全 193 班
实验类型：□ 验证 □ 综合 □ 设计 □ 创新　实验日期：2022.4.17　实验成绩：

一、实验项目名称

数字指纹算法的设计

二、实验目的

了解数字指纹算法的实现方式和指纹嵌入算法

三、实验基本原理

作为信息隐藏的一个重要分支，数字指纹主要用于版权保护。数字指纹是将不同的标志性识别代码——指纹，利用数字水印技术嵌入到数字媒体中，然后将嵌入了指纹的数字媒体分发给用户。发行商发现盗版行为后，就能通过提取盗版产品中的指纹，确定非法复制的来源，对盗版者进行起诉，从而起到版权保护的作用。

我们先利用 sha256 算法将学号姓名转成 256 位的哈希值，称作哈希摘要，然后将其嵌入图像中。

四、主要仪器设备及耗材

Windows11， pycharm，图片

五、实验步骤

一、Sha256Hash 算法生成哈希摘要

选取 sha256 算法作为指纹的编码算法

```
class SHA256:
    def __init__(self):
        #64 个常量
        #图中 Kt
        self.constants = (
            0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5,
            0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
            0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3,
            0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
            0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc,
            0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
            0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7,
            0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
            0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13,
            0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
            0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3,
            0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
            0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5,
            0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
            0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208,
            0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2)
        #迭代初始值，h0,h1,...,h7
        self.h = (
            0x6a09e667, 0xbb67ae85, 0x3c6ef372, 0xa54ff53a,
            0x510e527f, 0x9b05688c, 0x1f83d9ab, 0x5be0cd19)

    #x 循环右移 b 个 bit
    #rightrotate b bit
    def rightrotate(self, x, b):
        return ((x >> b) | (x << (32 - b))) & ((2**32)-1)

    #信息预处理。附加填充和附加长度值
    def Pad(self, W):
        return bytes(W, "ascii") + b"\x80" + (b"\x00" * ((55 if (
len(W) % 64) < 56 else 119) - (len(W) % 64))) + (
            (len(W) << 3).to_bytes(8, "big"))

    def Compress(self, Wt, Kt, A, B, C, D, E, F, G, H):
```

```
        return ((H + (self.rightrotate(E, 6) ^ self.rightrotate(E, 11) ^ self.rightrotate(E, 25)) + (
                    (E & F) ^ (~E & G)) + Wt + Kt) + (
                            self.rightrotate(A, 2) ^ self.rightrotate(A, 13) ^ self.rightrotate(A, 22)) + (
                            (A & B) ^ (A & C) ^ (B & C))) & ((2**32)-1), A, B, C, (D + (
                     H + (self.rightrotate(E, 6) ^ self.rightrotate(E, 11) ^ self.rightrotate(E, 25)) + (
                         (E & F) ^ (~E & G)) + Wt + Kt)) & ((2**32)-1), E, F, G

    def hash(self, message):
        message = self.Pad(message)
        digest = list(self.h)

        for i in range(0, len(message), 64):
            S = message[i: i + 64]
            W = [int.from_bytes(S[e: e + 4], "big") for e in range(0, 64, 4)] + ([0] * 48)

            #构造 64 个 word
            for j in range(16, 64):
                W[j] = (W[j - 16] + (
                            self.rightrotate(W[j - 15], 7) ^ self.rightrotate(W[j - 15], 18) ^ (W[j - 15] >> 3)) + W[
                            j - 7] + (self.rightrotate(W[j - 2], 17) ^ self.rightrotate(W[j - 2], 19) ^ (
                            W[j - 2] >> 10))) & ((2**32)-1)

            A, B, C, D, E, F, G, H = digest

            for j in range(64):
                A, B, C, D, E, F, G, H = self.Compress(W[j], self.constants[j], A, B, C, D, E, F, G, H)

        return "".join(format(h, "02x") for h in b"".join(
            d.to_bytes(4, "big") for d in [(x + y) & ((2**32)-1) for x, y in zip(digest, (A, B, C, D, E, F, G, H))]))


def main():
    encoder = SHA256()

```

```
    while True:
        message = input("Enter string: ")
        print(f"Output: {encoder.hash(message)}\n")


if __name__ == "__main__":
    main()
```

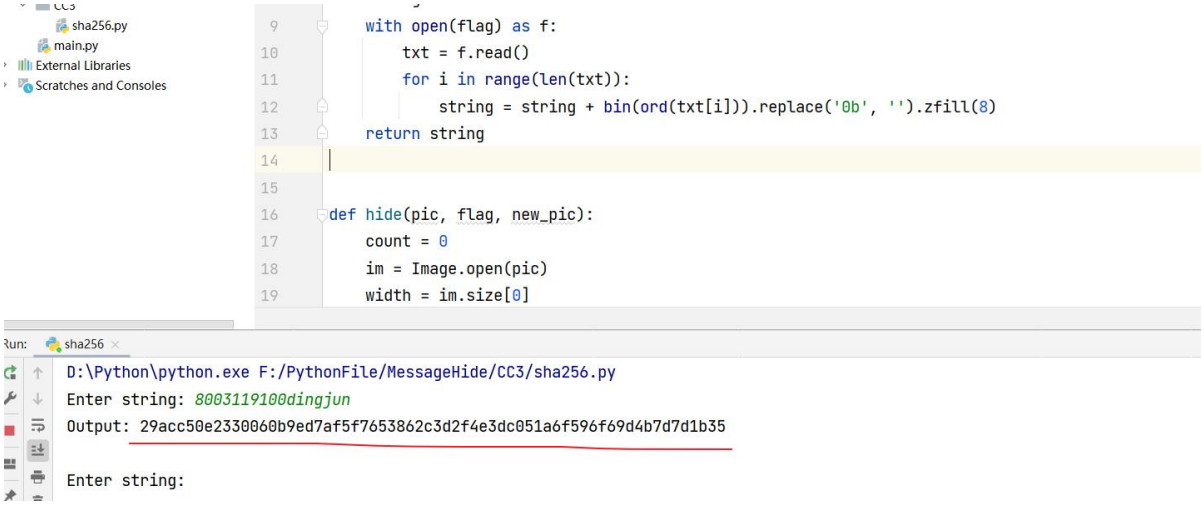
二、利用 LSB 数字水印 嵌入算法将生成的 256 位哈希摘要嵌入图像

```
from PIL import Image
def mod(x, y):
    return x % y

def bin_ord(message):
    string = ""
    txt = message
    for i in range(len(txt)):
        string = string + bin(ord(txt[i])).replace('0b', '').zfill(8)
    return string


def hide(pic, flag, new_pic):
    count = 0
    im = Image.open(pic)
    width = im.size[0]
    height = im.size[1]
    string = bin_ord(flag)
    for h in range(height):
        for w in range(width):
            pixel = im.getpixel((w, h))
            x = pixel[0]
            y = pixel[1]
            z = pixel[2]
            if count == len(string):
                break
            x = x - mod(x, 2) + int(string[count])
            im.putpixel((w, h), (x, y, z))
            count = count + 1
            if count == len(string):
```

```
                break;
            y = y - mod(y, 2) + int(string[count])
            im.putpixel((w, h), (x, y, z))
            count = count + 1
            if count == len(string):
                break
            z = z - mod(z, 2) + int(string[count])
            im.putpixel((w, h), (x, y, z))
            count = count + 1
    im.save(new_pic)


pic = 'pic.png'
new_pic = 'newpic.png'
flag = input("Enter hiding message:")
hide(pic, flag, new_pic)
```

六、实验数据及处理结果

输入我的学号姓名全屏执行 sha256 哈希函数生成 256 位消息摘要，当作要嵌入图像的指纹数据。



执行 LSB 数字水印嵌入算法，将数据指纹嵌入图像。

pic.png　　newpic.png

七、思考讨论题或体会或对改进实验的建议

可以考虑使用其他的数字指纹编码和嵌入算法

八、参考资料

信息隐藏技术