



# 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

```
1. class SHA256:
2.     def __init__(self):
3.         #64 个常量
4.         #图中 Kt
5.         self.constants = (
6.             0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5,
7.             0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
8.             0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3,
9.             0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
10.            0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc,
11.            0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
12.            0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7,
13.            0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
14.            0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13,
15.            0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
16.            0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3,
17.            0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
18.            0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5,
19.            0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
20.            0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208,
21.            0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2)
22.         #迭代初始值, h0,h1,...,h7
23.         self.h = (
24.             0x6a09e667, 0xbb67ae85, 0x3c6ef372, 0xa54ff53a,
25.             0x510e527f, 0x9b05688c, 0x1f83d9ab, 0x5be0cd19)
26.
27.         #x 循环右移 b 个 bit
28.         #rightrotate b bit
29.         def rightrotate(self, x, b):
30.             return ((x >> b) | (x << (32 - b))) & ((2**32)-1)
31.
32.         #信息预处理。附加填充和附加长度值
33.         def Pad(self, W):
34.             return bytes(W, "ascii") + b"\x80" + (b"\x00" * ((55 if (
35.                 len(W) % 64) < 56 else 119) - (len(W) % 64))) + (
36.                 (len(W) << 3).to_bytes(8, "big"))
37.         def Compress(self, Wt, Kt, A, B, C, D, E, F, G, H):
```

```

38.         return ((H + (self.rightrotate(E, 6) ^ self.rightrotate(E,
39.             11) ^ self.rightrotate(E, 25)) + (
40.                 (E & F) ^ (~E & G)) + Wt + Kt) + (
41.                     self.rightrotate(A, 2) ^ self.rightro
42.                         tate(A, 13) ^ self.rightrotate(A, 22)) + (
43.                             (A & B) ^ (A & C) ^ (B & C))) & ((2**
44.                                 32)-1), A, B, C, (D + (
45.                                     H + (self.rightrotate(E, 6) ^ self.rightrotat
46.                                         e(E, 11) ^ self.rightrotate(E, 25)) + (
47.                                             (E & F) ^ (~E & G)) + Wt + Kt)) & ((2**32)
48.                                                 -1), E, F, G
49.
50.     def hash(self, message):
51.         message = self.Pad(message)
52.         digest = list(self.h)
53.
54.         for i in range(0, len(message), 64):
55.             S = message[i: i + 64]
56.             W = [int.from_bytes(S[e: e + 4], "big") for e in rang
57.                 e(0, 64, 4)] + ([0] * 48)
58.
59.             #构造 64 个 word
60.             for j in range(16, 64):
61.                 W[j] = (W[j - 16] + (
62.                     self.rightrotate(W[j - 15], 7) ^ self.
63.                         rightrotate(W[j - 15], 18) ^ (W[j - 15] >> 3)) + W[
64.                             j - 7] + (self.rightrotate(W[j - 2],
65.                                 17) ^ self.rightrotate(W[j - 2], 19) ^ (
66.                                     W[j - 2] >> 10))) & ((2**32)-1)
67.
68.             A, B, C, D, E, F, G, H = digest
69.
70.             for j in range(64):
71.                 A, B, C, D, E, F, G, H = self.Compress(W[j], self.
72.                     constants[j], A, B, C, D, E, F, G, H)
73.
74.         return "".join(format(h, "02x") for h in b"".join(
75.             d.to_bytes(4, "big") for d in [(x + y) & ((2**32)-1)
76.             for x, y in zip(digest, (A, B, C, D, E, F, G, H))]))
77.
78. def main():
79.     encoder = SHA256()
80.

```

```
72.     while True:
73.         message = input("Enter string: ")
74.         print(f"Output: {encoder.hash(message)}\n")
75.
76.
77. if __name__ == "__main__":
78.     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

```
1.  from PIL import Image
2.  def mod(x, y):
3.      return x % y
4.
5.  def bin_ord(message):
6.      string = ""
7.      txt = message
8.      for i in range(len(txt)):
9.          string = string + bin(ord(txt[i])).replace('0b', '').zfill(8)
10.     return string
11.
12.
13. def hide(pic, flag, new_pic):
14.     count = 0
15.     im = Image.open(pic)
16.     width = im.size[0]
17.     height = im.size[1]
18.     string = bin_ord(flag)
19.     for h in range(height):
20.         for w in range(width):
21.             pixel = im.getpixel((w, h))
22.             x = pixel[0]
23.             y = pixel[1]
24.             z = pixel[2]
25.             if count == len(string):
26.                 break
27.             x = x - mod(x, 2) + int(string[count])
28.             im.putpixel((w, h), (x, y, z))
29.             count = count + 1
30.         if count == len(string):
```

```

31.             break;
32.             y = y - mod(y, 2) + int(string[count])
33.             im.putpixel((w, h), (x, y, z))
34.             count = count + 1
35.             if count == len(string):
36.                 break
37.             z = z - mod(z, 2) + int(string[count])
38.             im.putpixel((w, h), (x, y, z))
39.             count = count + 1
40.         im.save(new_pic)
41.
42.
43.     pic = 'pic.png'
44.     new_pic = 'newpic.png'
45.     flag = input("Enter hiding message:")
46.     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。



The screenshot shows a Python IDE with a file named `sha256.py` open. The code defines a `hide` function and a script to read a message from a file and generate a SHA256 hash. The output of the script is displayed in the console.

```

9         with open(flag) as f:
10             txt = f.read()
11             for i in range(len(txt)):
12                 string = string + bin(ord(txt[i])).replace('0b', '').zfill(8)
13             return string
14
15
16 def hide(pic, flag, new_pic):
17     count = 0
18     im = Image.open(pic)
19     width = im.size[0]

```

Run: sha256

D:\Python\python.exe F:/PythonFile/MessageHide/CC3/sha256.py

Enter string: 8003119100dingjun

Output: 29acc50e2330060b9ed7af5f7653862c3d2f4e3dc051a6f596f69d4b7d7d1b35

Enter string:

执行 LSB 数字水印嵌入算法，将数据指纹嵌入图像。



```
45     if count == len(string):
46         break
47     z = z - mod(z, 2) + int(string[count])
48     im.putpixel((w, h), (x, y, z))
49     count = count + 1
50 im.save(new_pic)
51
52
53 pic = 'pic.png'
54 new_pic = 'newpic.png'
55 flag = input("Enter hiding message:")
hide()
for h in range(height):
    for w in range(width):
        if count == len(string):
```

Run: sha256 x LSB x

D:\Python\python.exe F:/PythonFile/MessageHide/CC1/LSB.py

Enter hiding message:29acc50e2330060b9ed7af5f7653862c3d2f4e3dc051a6f596f69d4b7d7d1b35

Process finished with exit code 0



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