

密码学实验实验报告十

18374480-黄翔

2021 年 6 月 15 日

1 实验目的

1. 理解密码学 Hash 函数的安全性质、攻击方法、设计思想, 掌握常用的密码学 Hash 函数的实现流程。
2. 理解消息认证码的设计思想, 掌握基于 Hash 函数的消息认证码 HMAC 的实现方法。

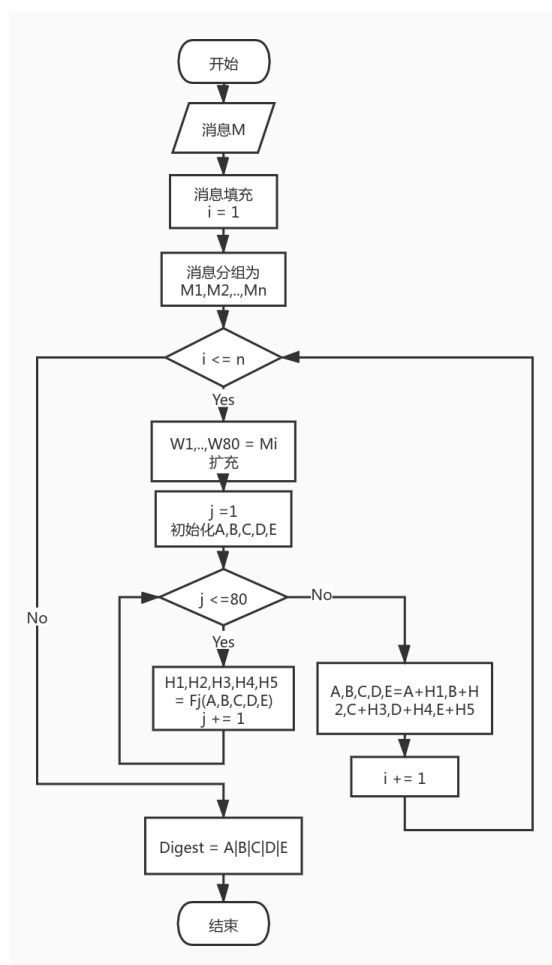
2 实验环境

1. python 3.9

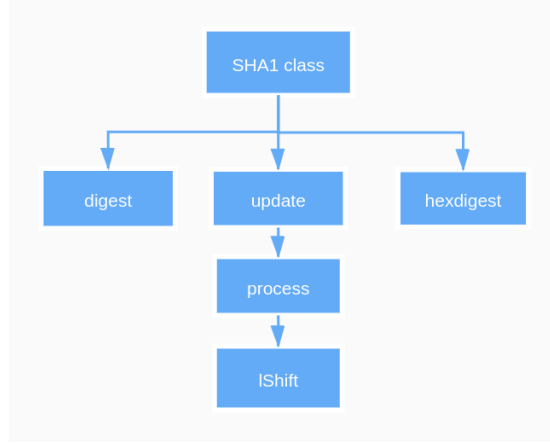
3 实验内容

3.1 SHA-1 hash 算法

3.1.1 算法流程图



3.1.2 函数调用关系



3.1.3 算法伪代码

Algorithm 1 SHA-1 hash

Input: 消息 $Message$

Output: 消息摘要 $Digest$

```

1:  $Message \leftarrow$  对  $Message$  消息填充
2:  $M_0, \dots, M_n \leftarrow$  对明文  $Message$  以 512bits 分组
3:  $A \leftarrow 67452301$ 
4:  $B \leftarrow EFCDAB89$ 
5:  $C \leftarrow 98BADCFE$ 
6:  $D \leftarrow 10325476$ 
7:  $E \leftarrow C3D2E1F0$ 
8: for  $i \leftarrow 0$  to  $n$  do
9:    $W_0, \dots, W_{15} \leftarrow M_i$ 
10:  for  $j \leftarrow 16$  to 79 do
11:     $W_j \leftarrow (W_{j-3} \oplus W_{j-8} \oplus W_{j-14} \oplus W_{j-16}) \lll 1$ 
12:  end for
13:   $H_0, H_1, H_2, H_3, H_4 \leftarrow A, B, C, D, E$ 
14:  for  $j \leftarrow 0$  to 79 do
15:    if  $0 \leq j < 20$  then
16:       $f \leftarrow (H_1 \& H_2) | (-H_1 \& H_3)$ 
17:       $k \leftarrow 5A827999$ 

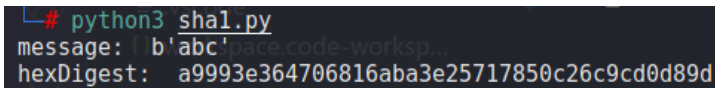
```

```

18:     end if
19:     if  $20 \leq j < 40$  then
20:          $f \leftarrow H_1 \oplus H_2 \oplus H_3$ 
21:          $k \leftarrow 6ED9EBA1$ 
22:     end if
23:     if  $40 \leq j < 60$  then
24:          $f \leftarrow (H_1 \& H_2) | (H_1 \& H_3) | (H_2 \& H_3)$ 
25:          $k \leftarrow 8F1BBCDC$ 
26:     end if
27:     if  $60 \leq j < 80$  then
28:          $f \leftarrow H_1 \oplus H_2 \oplus H_3$ 
29:          $k \leftarrow CA62C1D6$ 
30:     end if
31:      $H_4 \leftarrow H_3$ 
32:      $H_3 \leftarrow H_2$ 
33:      $H_2 \leftarrow H_1 \lll_{32} 30$ 
34:      $H_1 \leftarrow H_0$ 
35:      $H_0 \leftarrow H_0 \lll_{32} 5 + f + H_4 + W_j + k$ 
36: end for  $A, B, C, D, E \leftarrow H_0, H_1, H_2, H_3, H_4$ 
37: end for
38: return  $Digest \leftarrow (A|B|C|D|E)$ 

```

3.1.4 测试样例及结果截图



```

# python3 sha1.py
message: b'abc'
hexdigest: a9993e364706816aba3e25717850c26c9cd0d89d

```

3.1.5 总结

雪崩效应 hash 函数应该有良好的雪崩效应，一位的改变引起结果摘要的巨大变化

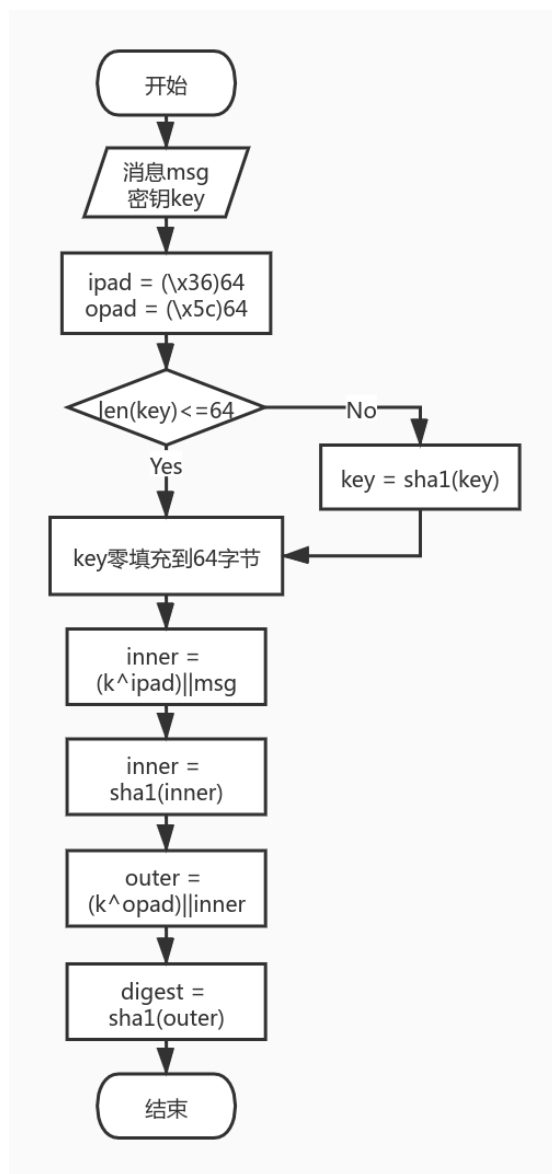
消息	消息摘要
abc	a9993e364706816aba3e25717850c26c9cd0d89d
abd	cb4cc28df0fdb0ecf9d9662e294b118092a5735

表 1: sha1 雪崩效应

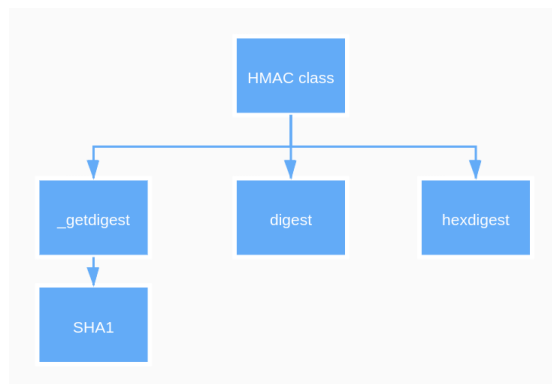
编写规范 因为需要在 HMAC 以及之后生日攻击中使用, 为了方便与 hashlib 库中 sha1 替换, 设计 sha1 类的使用类似 hashlib 中的使用方法, 留出 update, digest, hexdigest 三个函数接口供调用

3.2 HMAC-SHA1

3.2.1 算法流程图



3.2.2 函数调用关系



3.2.3 算法伪代码

Algorithm 2 HMAC-SHA1

Input: 消息 msg , 密钥 key

Output: 消息认证码 $digest$

```
1:  $ipad \leftarrow (\backslash x36)_{64}$ 
2:  $opad \leftarrow (\backslash x5c)_{64}$ 
3: if  $len(key) > 64$  then
4:    $key \leftarrow sha1(key)$ 
5: end if
6:  $key \leftarrow (key || \backslash x00)_{64}$ 
7:  $inner \leftarrow key \oplus ipad$ 
8:  $inner \leftarrow sha1(inner || msg)$ 
9:  $outer \leftarrow key \oplus opad$ 
10:  $digest \leftarrow sha1(outer || inner)$ 
11: return  $digest$ 
```

3.2.4 测试样例及结果截图

```
# python3 hmac.py
msg: b'Sample message for keylen<blocklen'
key: b'\x00\x01\x02\x03\x04\x05\x06\x07\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13'
hexdigest: 4c99ff0cb1b31bd33f8431dbaf4d17fcd356a807
```

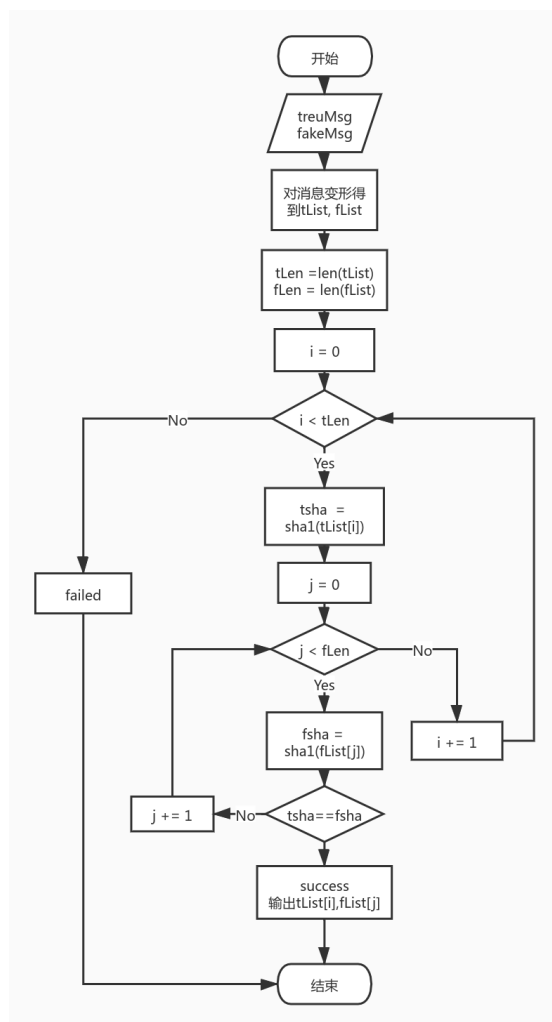
3.2.5 总结

安全性

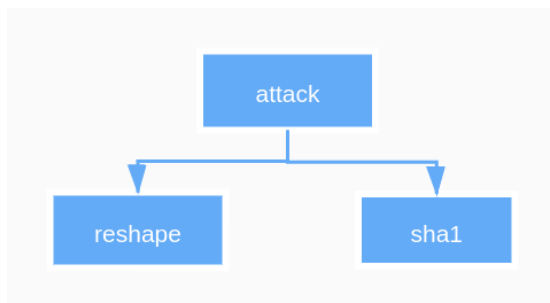
1. 用穷举方法确定认证密钥比确定同等长度的加密密钥更加困难
2. 攻击者构造出与 M 具有相同消息认证码的消息 M' 在计算上不可行
3. $MAC(K, M)$ 应是均匀分布的, 即对任何随机选择的消息 M 和 M' , $MAC(K, M) = MAC(K, M')$ 的概率是 2^{-n} , 其中 n 是 MAC 的位数
4. 设 M' 是 M 的某个已知变换, 即 $M' = f(M)$, 要求 $Pr[MAC(K, M) = MAC(K, M')] = 2^{-n}$

3.3 SHA1 第二类生日攻击

3.3.1 算法流程图



3.3.2 函数调用关系



3.3.3 算法伪代码

Algorithm 3 SHA1 第二类生日攻击

Input: 真消息 $trueMsg$, 假消息 $fakeMsg$, 攻击位数 $crackBits$

Output: $tAns, fAns$

```

1:  $tList[0 : 2^{\frac{crackBits}{2}}] \leftarrow$  对  $trueMsg$  变形
2:  $fList[0 : 2^{\frac{crackBits}{2}}] \leftarrow$  对  $fakeMsg$  变形'
3:  $tLen \leftarrow len(tList)$ 
4:  $fLen \leftarrow len(fList)$ 
5: for  $i \leftarrow 0$  to  $tLen - 1$  do
6:    $tsha \leftarrow sha1(tList[i])$ 
7:   for  $j \leftarrow 0$  to  $fLen - 1$  do
8:      $fsha \leftarrow sha1(fList[j])$ 
9:     if  $tsha[0 : crackBits - 1] = fsha[0 : crackBits - 1]$  then
10:      return  $tAns \leftarrow tList[i], fAns \leftarrow fList[j]$ 
11:    end if
12:  end for
13: end for
14: return  $failed!!!$ 

```

3.3.4 测试样例及结果截图

```

python3 birthdayAttack.py
reshape ok
b'h
b'fuck,
71d5433ea9cd01fc5fba40368984cd10fe5015c6
71d5a4ac7f4881fc42996e35e0111cc7c14dce0d
ello, world'
world'

```

3.3.5 总结

算法复杂度 对于 n 位的 hash 算法，需要穷举的次数为 $2^{\frac{n}{2}}$ ，碰撞成功的概率超过 0.5

攻击防范

1. 使用安全的 hash 算法：即产生的 hash 码位数足够大
2. 加盐：在为文件签名之前，先向文件添加一个随机值，然后计算哈希值，再将文件、签名和随机值一起发送给接收者。这样，攻击者必须找出具有特定哈希值的伪造文件，这非常困难。

4 总结

通过本次实验，我理解了密码学 Hash 函数的安全性质、攻击方法、设计思想，掌握常用的密码学 Hash 函数 SHA1 的实现流程。通过实现 HMAC-SHA1，我理解了消息认证码的设计思想，掌握基于 Hash 函数的消息认证码 HMAC 的实现方法。同时，通过实现第二类生日攻击，了解了这种碰撞攻击的思想，进一步理解 hash 算法的安全性