Experimental report——implement length extension attack for SM3

姓名：李祥方

学号：201900460041

完成时间：7月6日

1 前置知识

长度拓展攻击：在密码学和计算机安全中，长度扩展攻击（Length extension attacks）是指针对某些允许包含额外信息的加密散列函数的攻击手段。该攻击适用于在消息与密钥的长度已知的情形下，所有采取了H(密钥∥消息) 此类构造的散列函数。MD5和SHA-1等基于Merkle–Damgård构造的算法均对此类攻击显示出脆弱性。注意，由于密钥散列消息认证码（HMAC）并未采取H(密钥∥消息)的构造方式，因此不会受到此类攻击的影响（如HMAC-MD5、HMAC-SHA1）。SHA-3算法对此攻击免疫。 [1] 对此类攻击脆弱的散列函数的常规工作方式是：获取输入消息，利用其转换函数的内部状态；当所有输入均处理完毕后，由函数内部状态生成用于输出的散列摘要。因而存在着从散列摘要重新构建内部状态、并进一步用于处理新数据（攻击者伪造数据）的可能性。如是，攻击者得以扩充消息的长度，并为新的伪造消息计算出合法的散列摘要。

2 实验过程

根据以上前置知识，编写了以下攻击的代码，主要包括两个函数，**add_attack()的输入是原始消息的哈希值和长度，以及想添加的消息，另一个从上帝视角计算出原始数据加添加数据的正确哈希值用于对比:**

```python
def add_attack(h,ADD,LEN):
    #利用已知的LEN创造出正确的m_list
    m_list = SM3.fenzu(ADD)
    l = hex(len(ADD)*4 + LEN)[2:]
    l_len=len(l)
    m_list[-1]=m_list[-1][0:128-l_len]+l
    m_len = len(m_list)
    V = ['0' for i in range(m_len + 1)]
    V[0] = h
    for k in range(m_len):
        w = expand(m_list, k)
        W = w[0]
        W_0 = w[1]
        A = V[k][0:8]
        B = V[k][8:16]
        C = V[k][16:24]
        D = V[k][24:32]
        E = V[k][32:40]
        F = V[k][40:48]
        G = V[k][48:56]
        H = V[k][56:64]
        all = ''
        for j in range(64):
            b = a = SM3.Cyc_shift(A, 12)
            T = SM3.T_j(j)
            T = SM3.Cyc_shift(T, j)
            a = SM3.add(a, E)
            a = SM3.add(a, T)
            SS1 = SM3.Cyc_shift(a, 7)
            SS2 = SM3.or_16(SS1, b)
            b = SM3.FF_j(A, B, C, j)
            b = SM3.add(b, D)
            b = SM3.add(b, SS2)
            TT1 = SM3.add(b, W_0[j])  #
            b = SM3.GG_j(E, F, G, j)
            b = SM3.add(b, H)
            b = SM3.add(b, SS1)
            TT2 = SM3.add(b, W[j])  #
            D = C
            C = SM3.Cyc_shift(B, 9)
            B = A
            A = TT1  #
            H = G
            G = SM3.Cyc_shift(F, 19)
            F = E
            E = SM3.Replace_P0(TT2)  #
            all = A + B + C + D + E + F + G + H
        V[k + 1] = SM3.or_16(V[k], all)
    # print(V[-1])
    return V[-1]
```

```python
#以上帝视角先生成m+padding+ADD的hash,与add_attack对比判断结果是否一样,即可判断出攻击是否成功.

def known(m,ADD):
    after_fill = SM3.filling(m)+ADD
    outcome_know = SM3.hash(after_fill)
    return outcome_know

# 首先初始化原始的消息字
m = '12312'
# 默认设置消息字延长位
ADD = '12345'

#打印结果
#print(add_attack(SM3.hash(m),ADD,512*len(SM3.fenzu(m))))
#print(known(m,ADD))
if known(m,ADD) == add_attack(SM3.hash(m),ADD,512*len(SM3.fenzu(m))) :
    print('Successful implementation of length expansion attack')
else:
    print('Unsuccessful implementation of length expansion attack')
```

3 实验结果

初始化：

```python
# 首先初始化原始的消息字
m = '12312'
# 默认设置消息字延长位
ADD = '12345'
```

测试结果：

```
C:\Users\GL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/GL/Desktop/implement length extension attack for SM3/main.py"
['12345800000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000214']
Successful implementation of length expansion attack
```

由测试结果可知,128位最后三位为’214’,等于10进制中的532位，正好等于512+4*5=532位，根据打印的显示，可以判断代码运行正确.