山东大学网络空间安全学院

创新创业实践 课程实验报告

学号：202100141038	姓名：沈文慧	班级：21 密码 2 班
实验题目： implement the naïve birthday attack of reduced SM3		
实验学时：	实验日期：2023-07	

问题分析： implement the naïve birthday attack of reduced SM3

本实验要求利用生日攻击破解 SM3 算法找到 nbit 碰撞的原像，所以先随机生成一个 nbit 的字符串 str（这里只考虑数字字符串，其他字符串也可以转化到数字字符串），再用 sm3 计算其对应哈希值 H=H（sm3），如果我们只考虑前 n 比特（下文中的相等都是指前 n 比特），那么在前 0-2^n 范围内，至少存在一组碰撞的概率几乎是 100%。对于特定的消息 M，利用生日攻击去找到这个碰撞像可以任取两个消息 m1，m2，计算 h=SM3(abs(M1-M2))，直到得到 H= h。

硬件环境：Windows 11，X64

软件环境：visual studio code

实验步骤与内容：

1. 实验代码：

```
#include <iostream>
#include<stdlib.h>
using namespace std;

//定义碰撞长度
//int Collisionlen=0;
int cmphash(unsigned char* H1, unsigned char* H2, int Len)
{   //暂时不处理超过 32bit 的比较
    if (Len <= 32) { //取 int 比较
        uint a = *(int*)H1;
        uint b = *(int*)H2;
        uint mask = (int)pow(2, Collisionlen) - 1;
        if ((a & mask) == (b & mask))
            return 0;
    }
    return 1;
}

int Pollard_Rho(uint image, unsigned char* H, uint c, uint* preimage) //H=SM3(image)
{
    uint m1 = rand();
    uint m2 = m1;
```

```
    while (true)
    {
        m1 = F(m1, c);
        m2 =F(F(m2, c),c);
        if (m2 == m1)
            return 1;
        uint tmp = m2 - m1;
        string input = to_string(tmp).c_str();
        unsigned char output[SM3_OUTLEN];
        SM3(input, output);

        if (!cmphash(H, output, Collisionlen) && tmp != image) //如果碰撞了，就返回 0，同时打印了一下 sm3 值
        {
            *preimage = tmp;
            cout << "SM3(" << input << "):";
            print_Hashvalue(output, SM3_OUTLEN);
            return 0;
        }
    }
}

void PreimageAttack(uint image)
{
    uint preimage;
    string image_input = to_string(image);
    unsigned char image_output[SM3_OUTLEN];

    uint c = rand();
    while (Pollard_Rho(image, image_output, c, &preimage))
    {
        c = rand();
    }
}

int main()
{
    srand(time(NULL));//初始随机数会直接影响到找到环路的时间，好的时候只需要几秒，不好的时候要几分钟
    clock_t start, end;//定义 clock_t 变量
    start = clock();//开始时间
    unsigned int image = rand();
    PreimageAttack(image);
    end = clock();//结束时间
```

```
    cout << "花费时间：time=" << double(end - start) / CLOCKS_PER_SEC << "s" << endl;//输出时间（单位：s）
    return 0;
}
```

结论分析与体会：

假设存在 M ‘，有 SM3(M)=SM3(M’)，那么我们在前 2^n 比特范围内找到 M’的概率是 1/2^n,但是我们找两个数 m1 和 m2，满足 m1-m2 = M’的概率则是（C2n，2）/2^n,这个比较类似素数检测中的 Poll_rho 算法。 利用生日攻击找到任意一组前 n 比特的碰撞　　如果我们不是利用生日攻击去找碰撞像，而是单纯找两个消息 m1 和 m2 满足 SM3(m1)=SM3(m2)，那么最简单的肯定还是穷举，使用 hash 表，遇到碰撞就输出。

int cmphash(unsigned char* H1,unsigned char* H2,int Len)

因为我们只比较前几个 bit，因为大端序的问题，这些比特是倒着存放的，进行处理。

```
int cmphash(unsigned char* H1,unsigned char* H2,int Len)
{
    if(Len<=32){ //取int比较
        uint a = *(int*) H1;
        uint b = *(int*) H2;
        uint mask = (int)pow(2,Collisionlen)-1;
        if ((a&mask) == (b&mask))
            return 0;
    }
    return 1;   //暂时不处理超过32bit的比较
}
```

int Pollard_Rho(uint image,unsigned char* H,uint c,uint* preiamge)

经典的 pollard rho 算法，使用平方加 c 的方式进行套圈

```
int Pollard_Rho(uint image,unsigned char* H,uint c,uint* preiamge) //H = SM3(image)
{
    uint m1 = rand();
    uint m2 = m1;
    while(true)
    {
        m1 = F(m1,c);
        m2 = F(F(m2,c),c);
        if(m2==m1)
            return 1;
        uint tmp = m2-m1;
        string input = to_string(tmp).c_str();
        unsigned char output[SM3_OUTLEN];
        SM3(input,output);

        if(!cmphash(H,output,Collisionlen) && tmp!=image)   //如果碰撞了，就返回0，同时打印了一下sm3值
        {
            *preiamge = tmp;
            cout << "SM3("<<input<<"):";
            print_Hashvalue(output,SM3_OUTLEN);
            return 0;
        }
    }
}
```

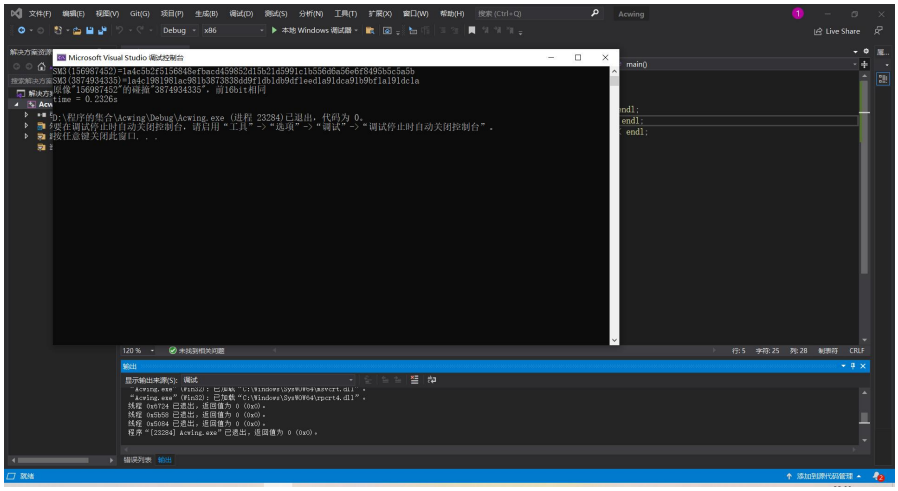
void PreimageAttack(uint image)

进行碰撞攻击。

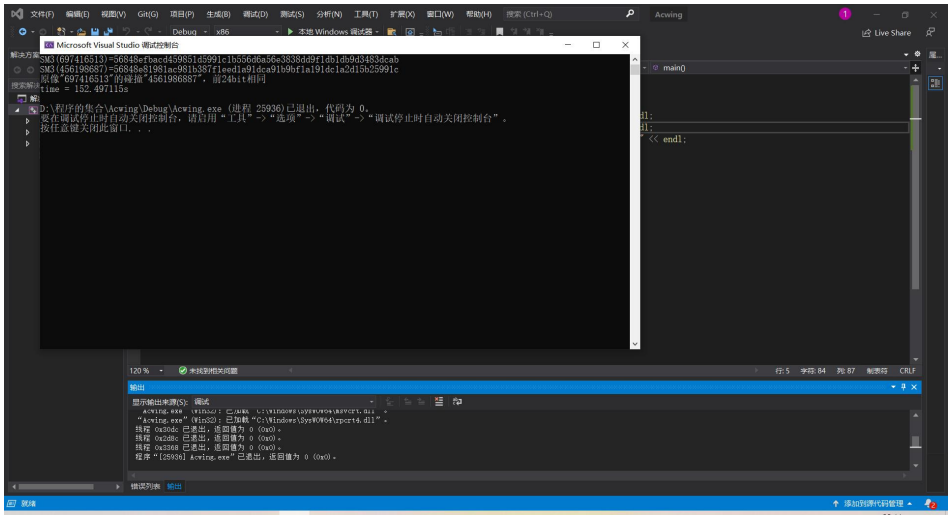
```
void  PreimageAttack(uint image)
{
    uint preimage;
    string image_input=to_string(image);
    unsigned char image_output[SM3_OUTLEN];
    SM3(image_input,image_output);
    cout << "SM3("<<image_input<<"):";
    print_Hashvalue(image_output,SM3_OUTLEN);

    uint c = rand();
    while(Pollard_Rho(image,image_output,c,&preimage))
    {
        c = rand();
    }
}
```

结果分析：

16bit 的原像碰撞很快。



24bit 明显变慢。



这符合我们的预期。