

birthday_attack项目说明

本项目是针对sm3前n位的生日攻击

代码说明

本代码使用openssl库的相关函数实现sm3的过程，通过随机枚举明文记录其散列值并比较的方式来尝试找到一对明文，其sm3后前n位的散列值相等。

其中值得注意的地方有：

1.在匹配散列值的过程中，使用了stl中的map容器，本质上是平衡树实现的字典，第一关键字是散列值，第二关键字是明文是第几次随机随机出来的（见后面明文存储相关），其相关代码如下：

```
//定义
map<unsigned int, map<unsigned int, int>> f;//64bit
//map<unsigned int, int> f;//32bit
//寻找是否有相同散列值
if (f.find(a) != f.end() && f[a].find(b) != f[a].end()) {
    printf("found %d,%d\n", f[a][b], t);//0 134217728
    break;
} //64bit
if (f.find(a) != f.end()) {
    printf("found %d,%d\n", f[a], t);//37626 92666
    break;
} //32bit
```

2.由于明文直接存储过于占据空间，所以此处利用固定随机种子后随机序列固定的特性，只需记录明文在随机序列中的位置，想要验证的时候只需重新设置随机种子，取出随机序列对应位置的明文即可。其相关代码如下：

```
if (i != x && i != y) {
    for (int i = 0; i < 16; i++) {
        rand();
    }
    continue;
} //当不是对应位置的时候直接不管
for (int i = 0; i < 16; i++) {
    *((int*)s + i) = rand();
} //当是对应位置的时候取出明文并hash
```

运行指导

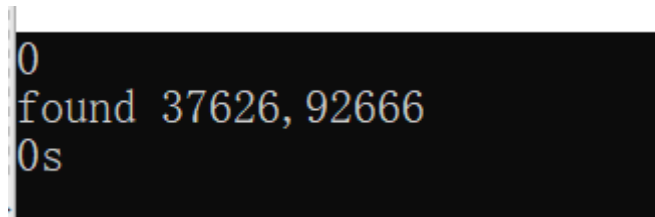
使用VS2019community，创建新项目，将源代码复制进去，在右侧设置项目属性，添加openssl库，即可运行。（因为使用了stl，建议开启O2优化）

想要实行生日攻击：设置CHECK常量=0，然后运行，就是前64bit的生日攻击，注释掉64bit的相关内容并反注释掉32bit的相关内容，则是前32bit的生日攻击。生日攻击的过程中会每进行一百万次hash出现一次计数，并在得到碰撞是输出两个明文对应的hash序列的位置。

想要验证：设置CHECK常量=1，然后运行，输入得到的两个位置，如果是32bit的比较，需要将main函数中的check_64bit(x,y)改成check_32bit(x,y)，若输出Right则说明其sm3散列值前n位相等。

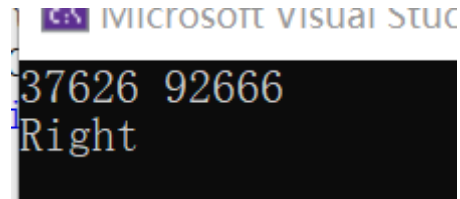
运行全过程截图

32bit生日攻击：



```
0
found 37626, 92666
0s
```

32bit验证：



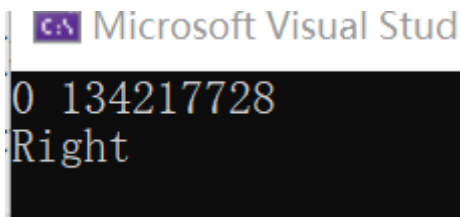
```
C:\> MICROSOFT VISUAL STU
37626 92666
Right
```

64bit生日攻击：

输出为0 134217728

(没时间跑了

64bit验证：



```
C:\> Microsoft Visual Stud
0 134217728
Right
```

贡献说明

本组只有一个人

参考

sm3的openssl实现参考了这篇博客https://blog.csdn.net/kyle_shaw/article/details/125052823