## 1.实现国密算法（SM3）生日攻击：

### 实现方式

###### 生日攻击

生日攻击是一种密码学攻击手段，所利用的是概率论中生日问题的数学原理。这种攻击手段可用于滥用两个或多个集团之间的通信。此攻击依赖于在随机攻击中的高碰撞概率和固定置换次数（鸽巢原理）。使用生日攻击，攻击者可在中找到散列函数碰撞，为原像抗性安全性。生日攻击是利用概率论中的生日问题，找到冲突的Hash值，伪造报文，使身份验证算法失效。

###### SM3

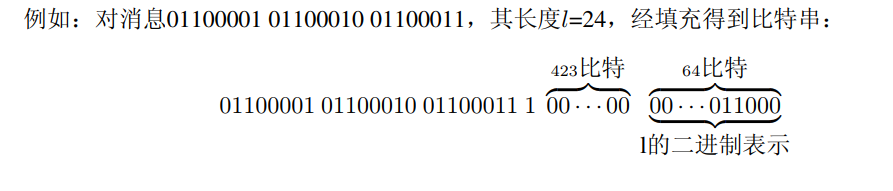
SM3密码杂凑算法是中国国家密码管理局2010年公布的中国商用密码杂凑算法标准。整个算法的执行过程可以概括成四个步骤：消息填充、消息扩展、迭代压缩、输出结果。

消息填充

SM3的消息扩展步骤是以512位的数据分组作为输入的。因此，我们需要在一开始就把数据长度填充至512位的倍数。数据填充规则和MD5一样，具体步骤如下：

1、先填充一个“1”，后面加上k个“0”。其中k是满足(n+1+k) mod 512 = 448的最小正整数。

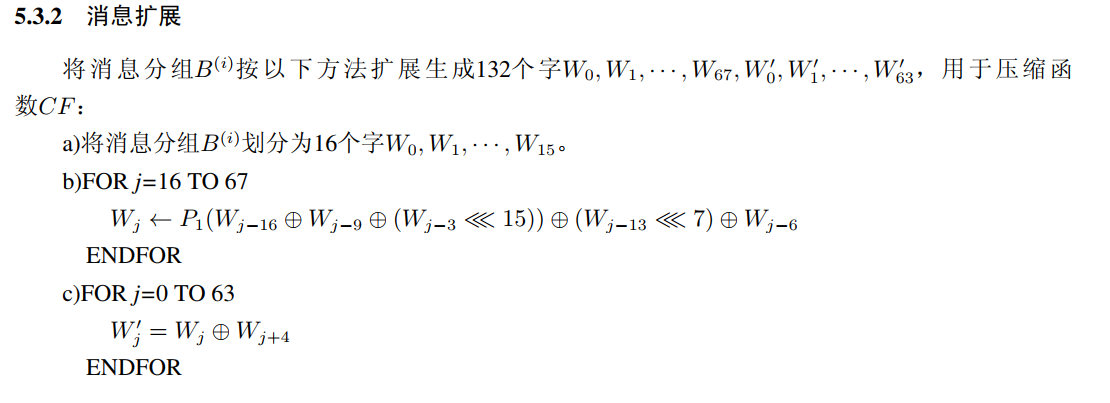
2、追加64位的数据长度（bit为单位，大端序存放[1](https://blog.csdn.net/qq_40662424/article/details/121637732#fn1)。观察算法标准原文附录A运算示例可以推知。）



消息扩展

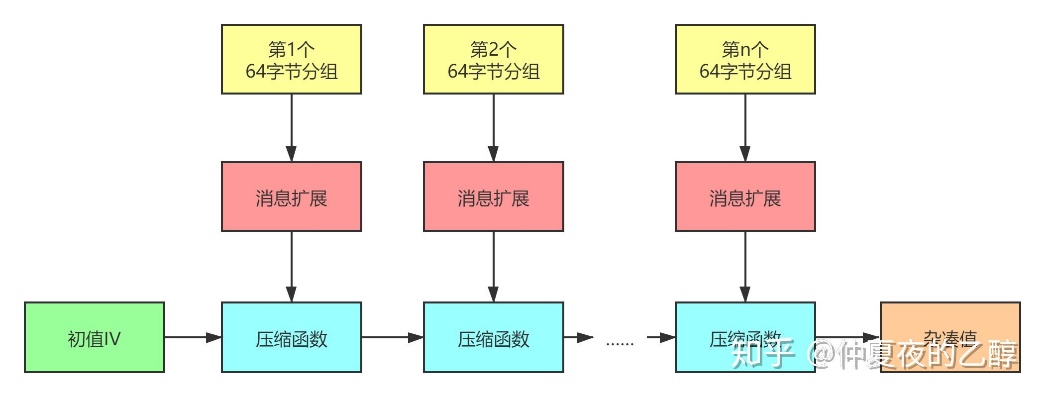
SM3的迭代压缩步骤没有直接使用数据分组进行运算，而是使用这个步骤产生的132个消息字。（一个消息字的长度为32位/4个字节/8个16j进制数字）概括来说，先将一个512位数据分组划分为16个消息字，并且作为生成的132个消息字的前16个。再用这16个消息字递推生成剩余的116个消息字。

在最终得到的132个消息字中，前68个消息字构成数列 {W j }，后64个消息字构成数列 {W j ‘​}，其中下标j从0开始计数。

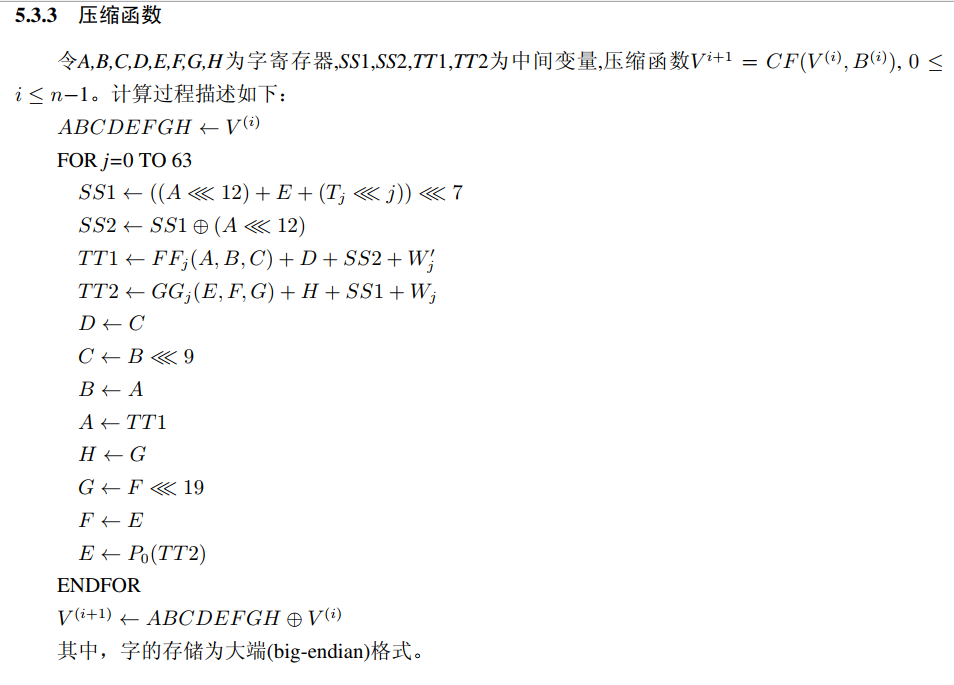
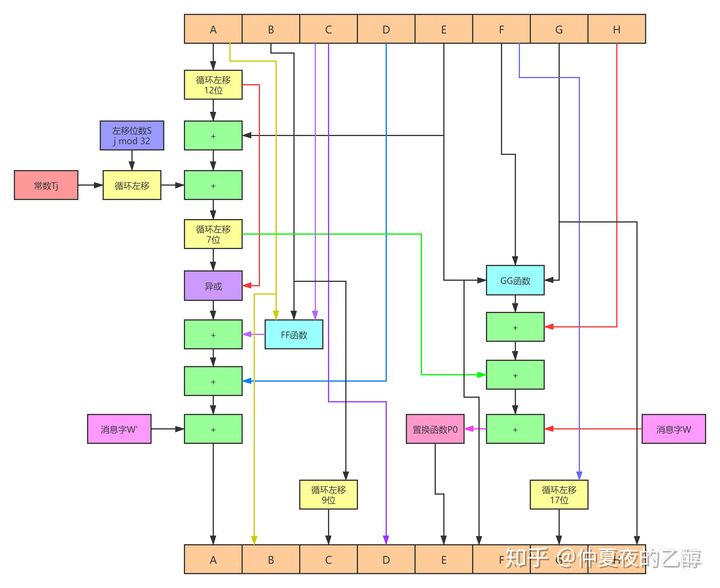


迭代压缩

在上文已经提过，SM3的迭代过程和MD5类似，也是Merkle-Damgard结构。但和MD5不同的是，**SM3使用消息扩展得到的消息字进行运算**。这个迭代过程可以用这幅图表示：



初值IV被放在A、B、C、D、E、F、G、H八个[32位变量](https://www.zhihu.com/search?q=32%E4%BD%8D%E5%8F%98%E9%87%8F&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra=%7B%22sourceType%22%3A%22article%22%2C%22sourceId%22%3A129692191%7D)中，其具体数值参见参考文献[1]。整个算法中最核心、也最复杂的地方就在于压缩函数。压缩函数将这八个变量进行64轮相同的计算，一轮的计算过程如下图所示：

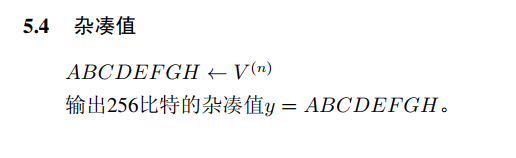


图中不同的数据流向用不同颜色的箭头表示。

最后，再将计算完成的A、B、C、D、E、F、G、H和原来的A、B、C、D、E、F、G、H分别进行异或，就是压缩函数的输出。这个输出再作为下一次调用压缩函数时的初值。依次类推，直到用完最后一组132个消息字为止。

输出结果

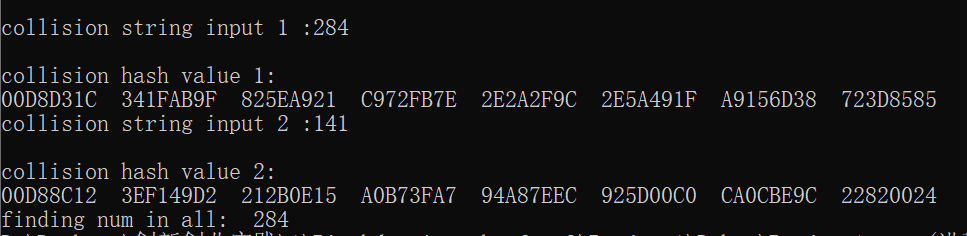
将得到的A、B、C、D、E、F、G、H八个变量拼接输出，就是SM3算法的输出。



###### 简要介绍

该次实验的SM3生日攻击是一定长度和hash值长度，在密文空间以50%以上的概率找到一个hash碰撞。通过使用查找表攻击的数据结构，我们可以选择在一边存表一边查表的情况下，于较短时间内找到一个16bit的hash弱碰撞

### 实现效果

****

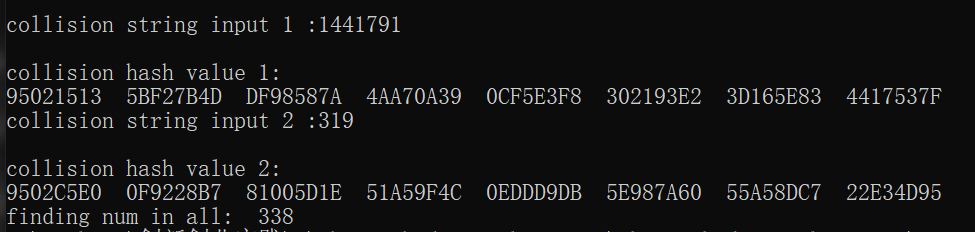
## 2.实现国密算法（SM3）ρ攻击：

### 实现方式

关于SM3的相关介绍见实验一实现方式，此处不再赘述

ρ攻击是通过ρ递推表达式来尝试寻求一个基于sm3哈希值的弱碰撞。 这通过使用查找表攻击的数据结构，可以在一边存表一边查表的情况下，于较短时间内找到一个16bit的hash弱碰撞。由结果可得，寻找16bit碰撞中生日攻击所需要的时间要少于ρ攻击，但若加长碰撞的bit长度，ρ攻击会比生日攻击展现出更好的效率。

### 实现效果

****

## 3.实现国密算法（SM3）长度扩展攻击：

### 实现方式

关于SM3的相关介绍见实验一实现方式，此处不再赘述

长度扩展攻击（length extension attack），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。对于满足以下条件的散列函数，都可以作为攻击对象：

① 加密前将待加密的明文按一定规则填充到固定长度（例如512或1024比特）的倍数；

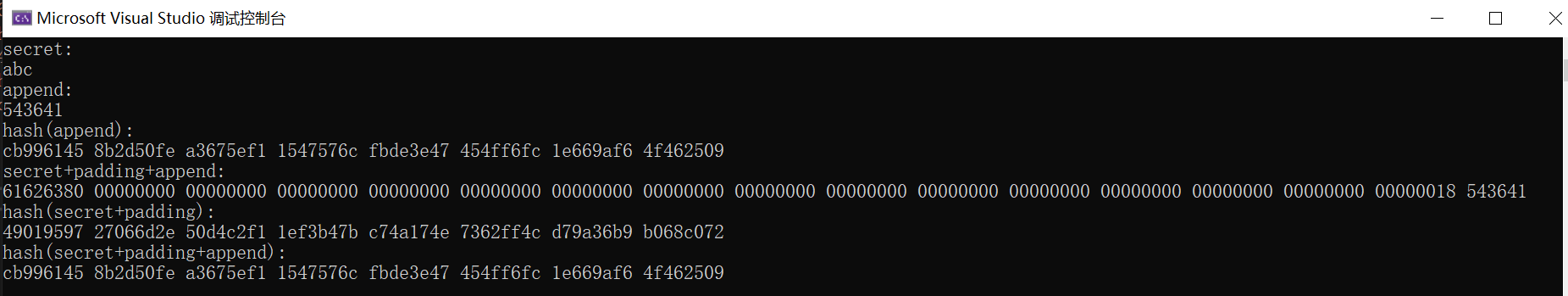
② 按照该固定长度，将明文分块加密，并用前一个块的加密结果，作为下一块加密的初始向量（Initial Vector）。

### 截图 2022-07-31 054733

在本实验中我们选择先随机生成一个消息M，再通过使用SM3函数算出其哈希值H。然后生成一个附加消息m。我们可以先通过H推算出该次加密结束后8个向量的值，再将其作为一个初始向量来加密m，这样我们即可得到另一个hash值H’

计算M + m+ padd的hash值H’’，若攻击成功，则H’应与H’’相等。

### 实现效果



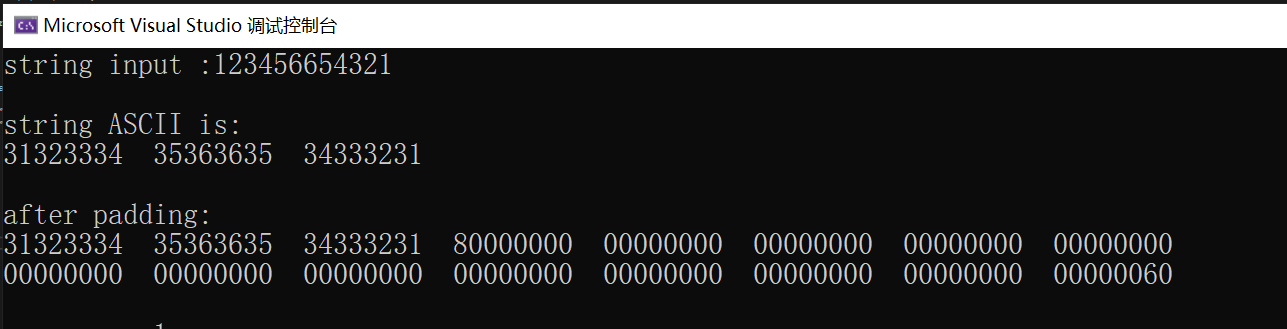
## 4.实现国密算法（SM3）软件层面优化：

### 实现方式

关于SM3的相关介绍见实验一实现方式，此处不再赘述

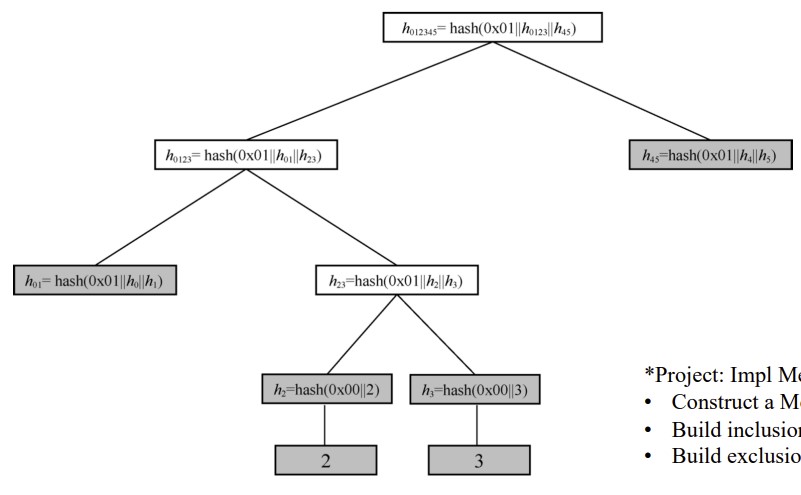
使用的做法比较基础，主要是通过运用在系统原理课程中学习到的知识，在循环展开等一些基础方面对软件层面实现优化加速。例如以下几种优化加速：并行加速、调换 循环顺序加速、SIMD 加速。

### 实现效果



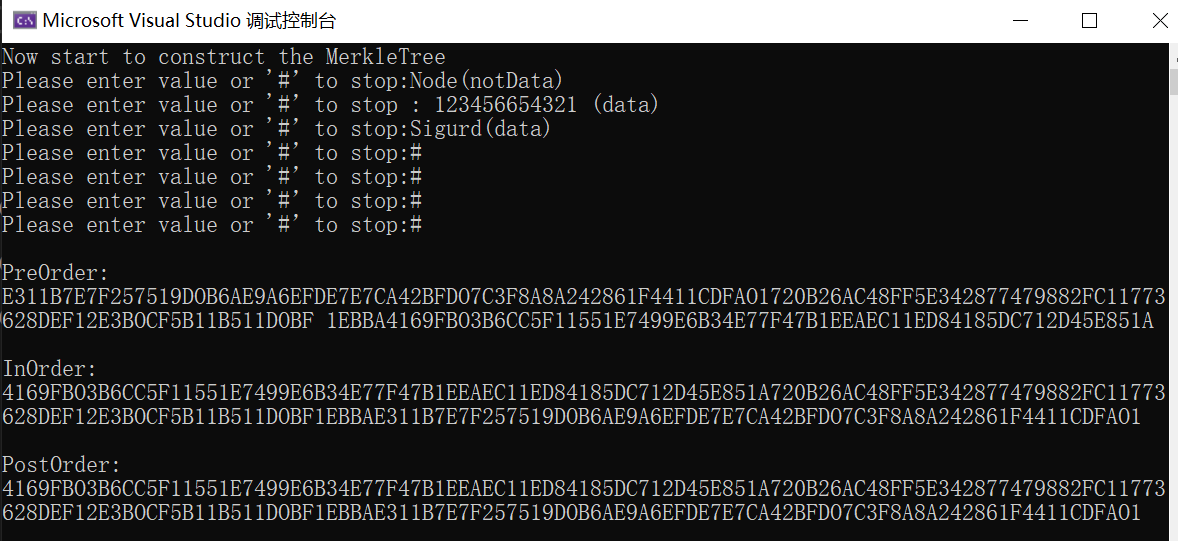
## 5.实现RFC6962的默克尔树：

### 实现方式



将sm3的散列模块作为基础，借助openssl库，使用C++实现手动输入数据，默克尔树叶节点生成相应的哈希值，非叶节点生成相应的联结哈希值。

### 实现效果



## 9. SM4软件层面实现：

### 实现方式

###### SM4

SM4算法是一个分组算法。数据分组长度为128比特，[密钥](https://so.csdn.net/so/search?q=%E5%AF%86%E9%92%A5&spm=1001.2101.3001.7020)长度为128比特。加密算法与密钥扩展都采用32轮迭代结构。SM4以字节（8比特）和字（32比特）位单位进行数据处理。它是对合算法，因此加解密算法相同，只是轮密钥的使用顺序相反，解密轮密钥是加密轮密钥的逆序。

1.基本运算

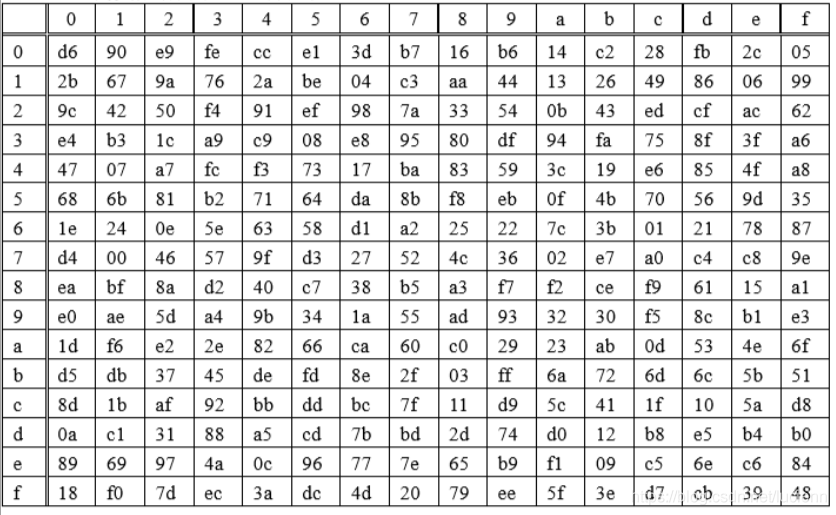
SM4算法使用模2加和循环移位作为基本运算。

2.基本密码部件

SM4密码算法使用了以下基本密码部件

（1）S盒

SM4的S盒是一种以字节为单位的非线性代替变换，其密码学的作用是起混淆作用。S盒的输入和输出都是8位的字节。它的本质上是8位的非线性置换。例如S盒输入为EF，则取第E行第F列交点处的值进行替换。

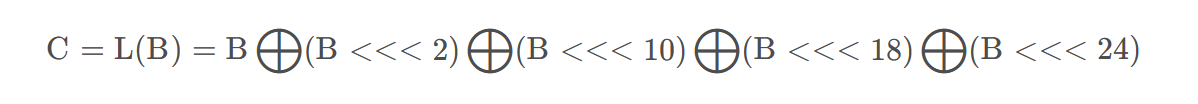


（2）非线性变换τ

SM4的非线性变换τ是一种以字为单位的非线性变换。它由4个S盒并置构成。这里的非线性变换就和之前zuc的S盒变换原理相同，一个字有4个字节，每个字节都使用S盒进行变化，再将结果进行首尾连接。不同的是ZUC使用的是两个S盒，这里只有一个S盒

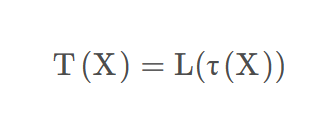
（3）线性变换L

线性变换L是以字为处理单位的线性变换部件，其输入输出都是32位的字。其密码学的作用是起扩散作用。设L的输入为字B，输出为字C，则：



（4）合成变换T

合成变换是由非线性变换τ和线性变换L复合而成。



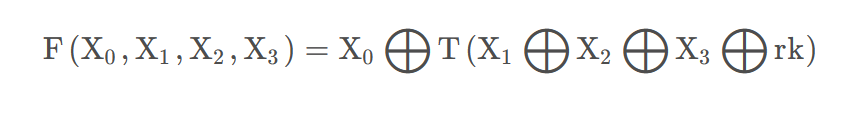
合成变换T起到混淆和扩散作用，提高密码安全性

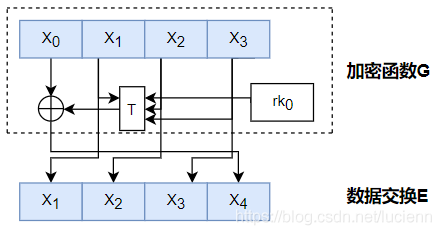
3.轮函数

SM4密码算法的轮函数是一种以字为处理单位的密码函数。

设轮函数F FF的输入为( X0 , X1 , X2 , X3 )

，四个32位字，共128位。轮密钥为rk，rk也是一个32位的字。其输出也是32位的字。





4.基本运算

SM4密码使用128位的加密密钥，并采用32轮迭代加密结构，每一轮加密使用一个32位的轮密钥，共使用32个轮密钥。因此需要使用密钥扩展算法。

（1）系统参数FK

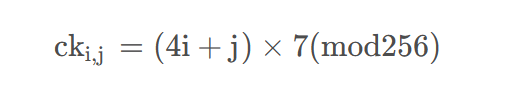
在密钥扩展中使用如下的参数：

FK0=(A3B1BAC6)，FK1=(56AA3350)，FK2=(677D9197)，FK3=(B27022DC)，

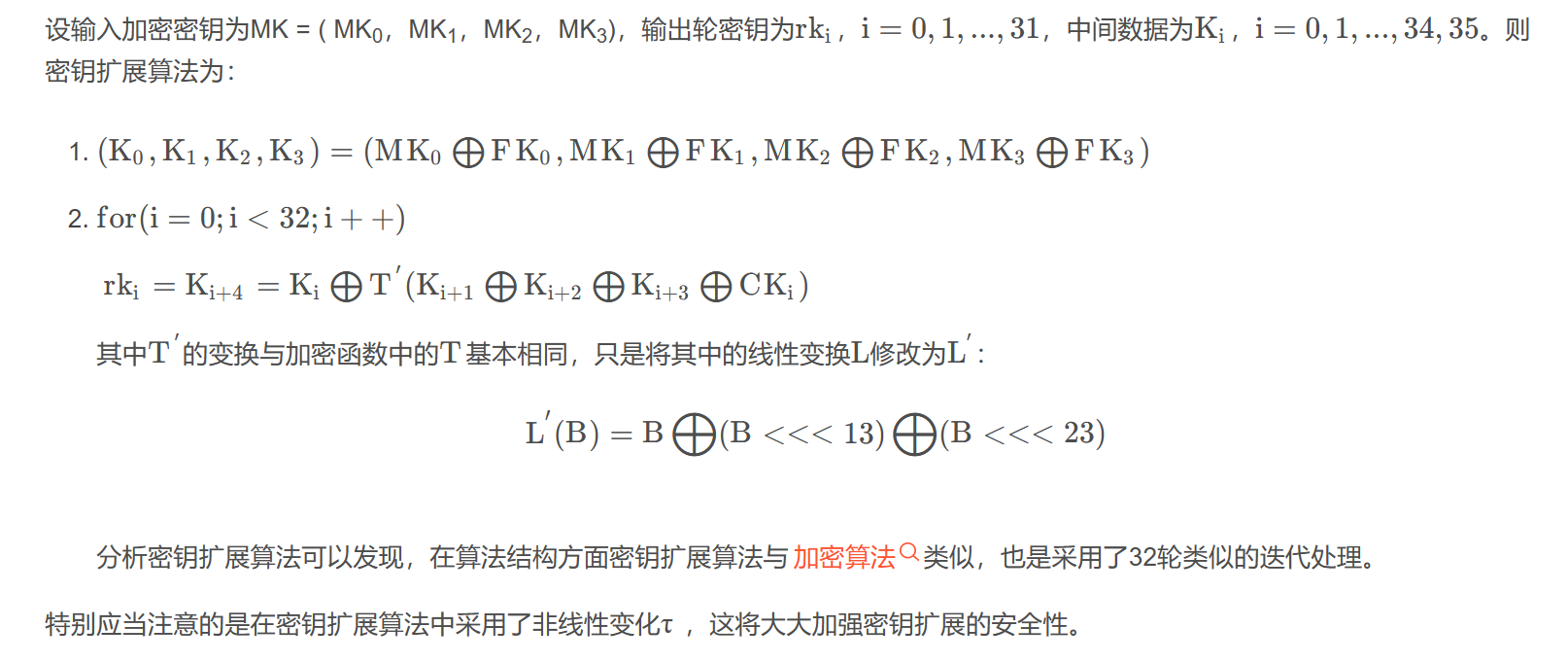
（2）固定参数CK

共使用32个固定参数CKi，CKi是一个字，其产生规则如下：

设cki,j 为CKi的第j字节( i = 0,1,…,31; j=0,1,2,3 )，即CKi = ( cki,0 , cki,1 , cki,2 , cki,3 )，则



（3）密钥扩展算法

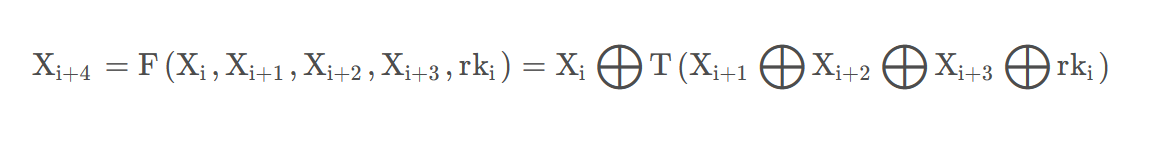


5.加密过程

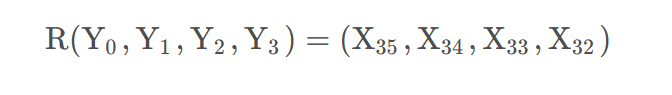
SM4密码算法是一个分组算法。数据分组长度为128比特，密钥长度为128比特。加密算法采用32轮迭代结构，每轮使用一个轮密钥。

设输入的明文为( X0 , X1 , X2 , X3 )，输入轮密钥为rki，i=0,1,..,31，共32个字。输出的密文为( Y0 , Y1 , Y2 , Y3 )

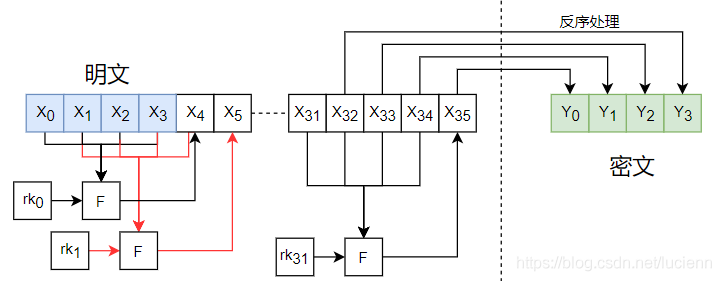
加密算法为：



为了与解密算法需要的顺序一致，在加密算法之后还需要一个反序处理R：



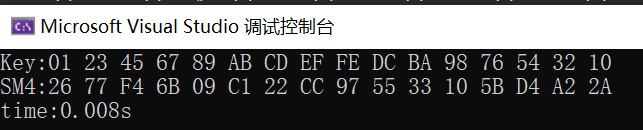
加密算法流程如下：



###### 简要介绍

本次实验我们通过调用和组装生成可以执行的sm4模块对sm4的结构进行了底层的函数实现，由于硬件环境有限且没有加速部件，所以运行起来可能相对比较慢。

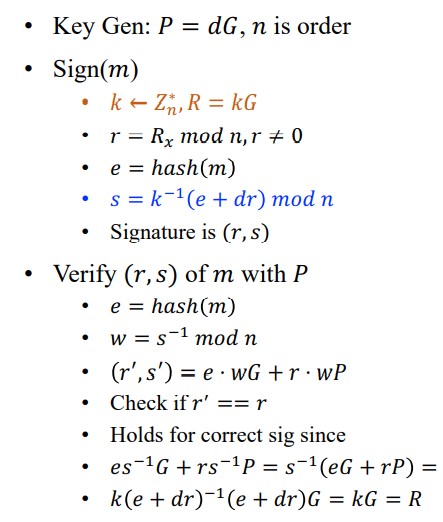
### 实现效果



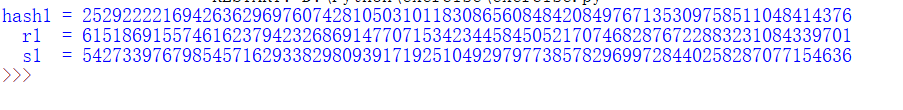
## 10. 伪造ECDSA签名：

### 实现方式

椭圆曲线数字签名算法（ECDSA）是使用椭圆曲线密码（ECC）对数字签名算法（DSA）的模拟。ECDSA是ECC与DSA的结合，整个签名过程与DSA类似，所不一样的是签名中采取的算法为ECC，最后签名出来的值也是分为r,s。本次实验我们通过在网上查阅参考资料的帮助下，使用python语言编写了一个简易的Sage脚本，可以伪造出基于ECDSA的合法签名。ECDSA的算法描述如下：



### 实现效果



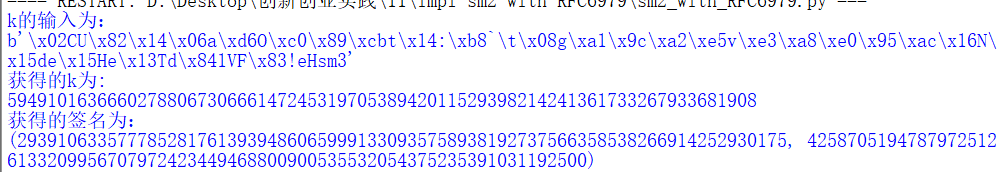
## 11. RFC6979实现SM2算法：

### 实现方式

由于k值泄露，两个人用了同样的k，或者两个同样的算法用了同样的k都可能导致私钥的泄露，所以RFC6979规定对与k这样的随机数的选取要变的不随机

在本次实验中我规定令k等于hash（d||ID||"算法类型"）其中d为私钥，ID为ID，"算法类型"为"sm3"

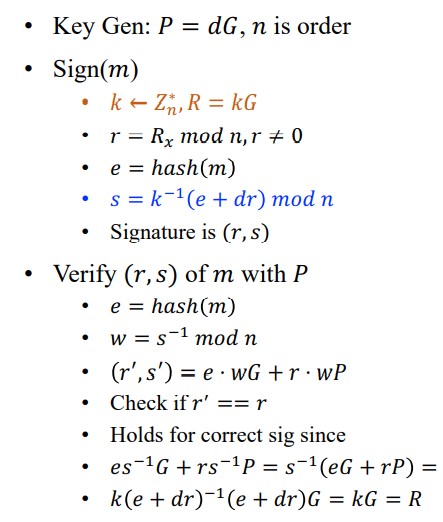
### 实现效果



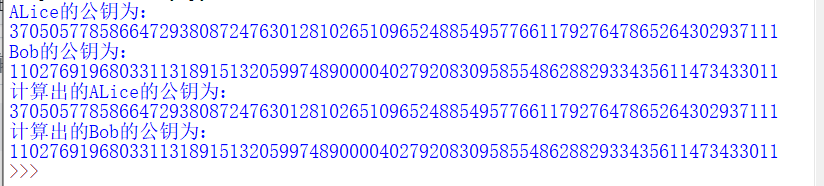
## 12. 使用概念验证代码验证上述陷阱：

### 实现方式

当A和B选择的K相同时，通过下面算法可以知道互相的私钥



### 实现效果

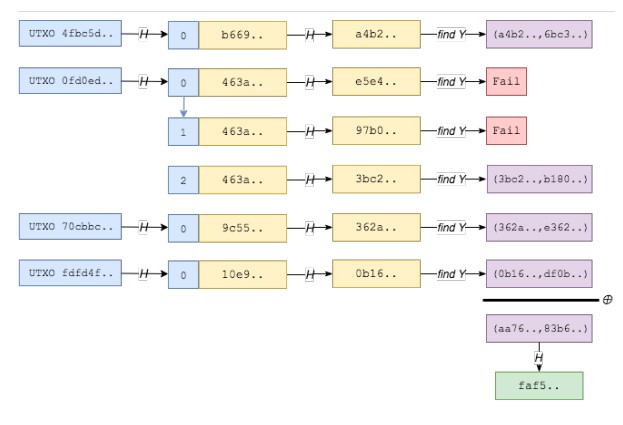


## 13. ECMH：

### 实现方式

在ECMH整体思路是：先把集合里的元素映射成椭圆曲线上的点，然后利用椭圆曲线上的加法求解哈希值。

为达到相同的安全性，ECMH算法需要的密钥长度远远小于哈希求和算法，因而ECMH相较哈希求和算法更为安全。



1.首先先对消息进行哈希

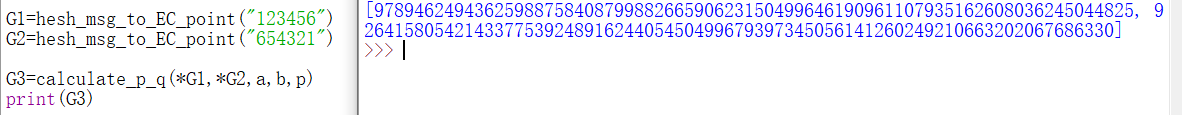
2.开始遍历从零到正无穷，并补充在hash之前得到hash2，对hash2做hash

3.令得到的hash值模p赋值给X

4.根据求二次剩余的算法得到y，如果有y就跳转到5，如果没有y就到2

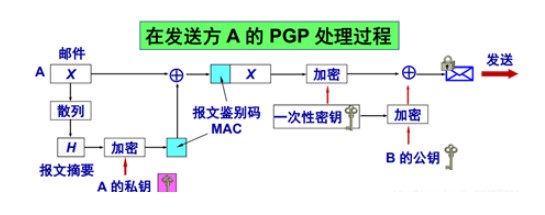
5.由于二次剩余得到的y有两个，那么这样选择两个y中的一个，根据消息的编码，如果最后以个数字为0，选小一点的y，如果为1，选大一点的y

### 实现效果



## 14. 带SM2的PGP方案：

### 实现方式

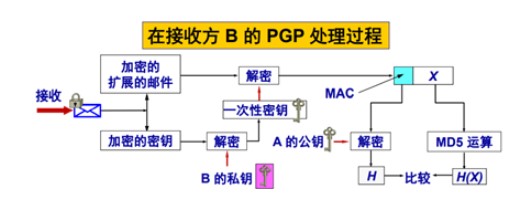


发送方：

1. 对明文邮件 X 进行 MD5 运算，得出 MD5 报文摘要 H。用 A 的私钥对 H 进行加密（即数字签名），得出报文鉴别码 MAC（即sig（H（M））），把它拼接在明文 X 后面，得到扩展的邮件 X ||麦克。

补：对该发送的消息进行压缩，记为Z（sig（H（M）） ||M)；

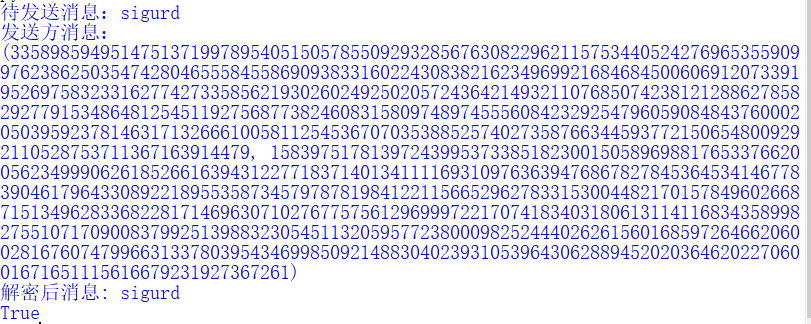
1. 使用 A 自己生成的一次性密钥Ks对扩展的邮件X ||MAC进行加密。
2. 用 B 的公钥对 A 生成的一次性密钥进行加密，即EB公钥（Ks）。因为加密所用的密钥是一次性的，即密钥只会使用一次，不会出现因为密钥泄露导致之前的加密内容被解密。即使密钥被泄露了，也只会影响一次通信过程。
3. 把加了密的一次性密钥和加了密的扩展的邮件连接（即EB公钥（Ks） ||EKs（Z（sig（H（M）） ||M））））发送给 B。



接收方：

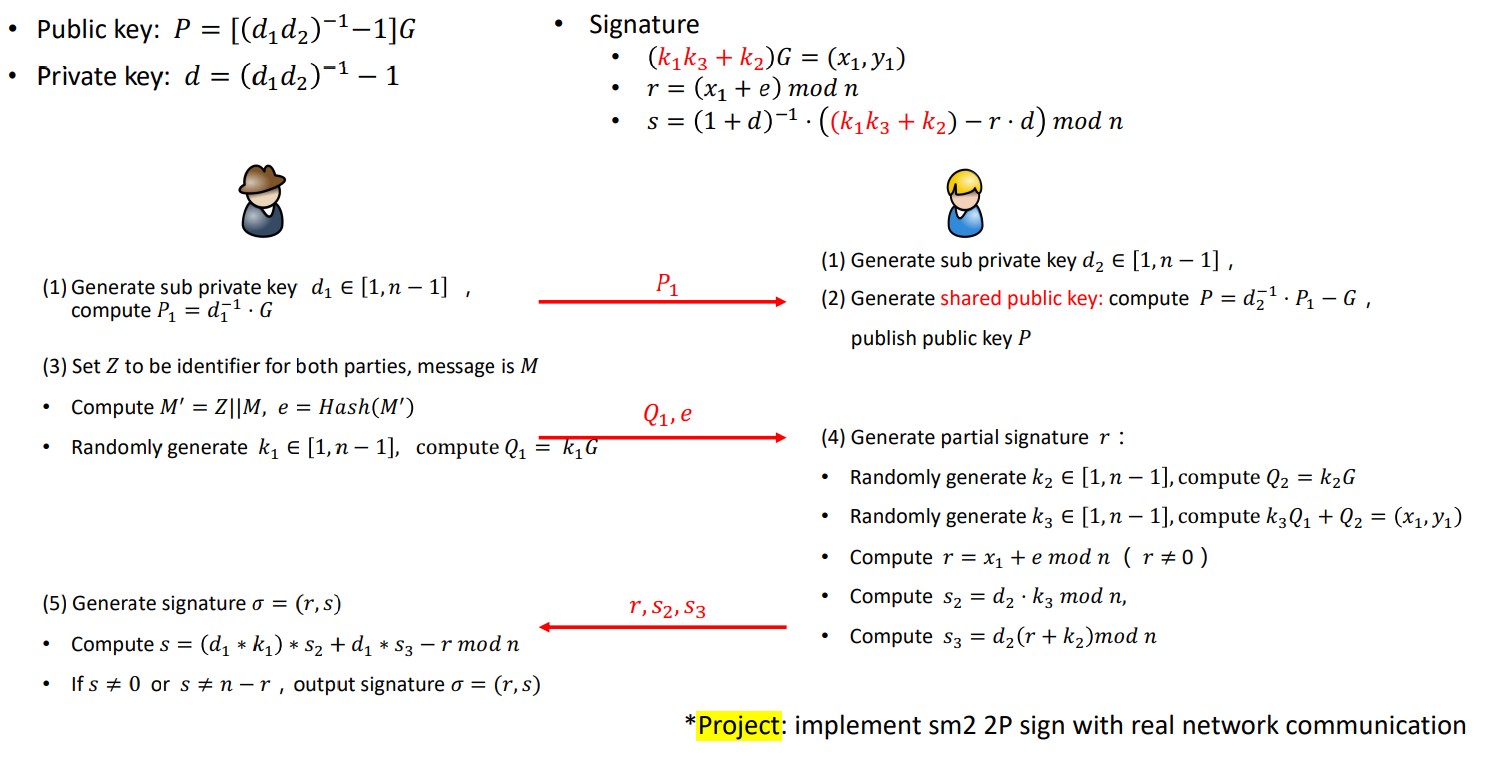
1. 把被加密的一次性密钥EB公钥（Ks）和被加密的扩展报文X ||MAC分离开。
2. 用 B 自己的私钥解出 A 的一次性密钥Ks。
3. 用解出的一次性密钥Ks对报文进行解密，然后分离出明文 X 和MAC。
4. 用 A 的公钥对 MAC 进行解密（即签名核实），得出报文摘要 H。这个报文摘要就是 A 原先用明文邮件 X 通过 MD5 运算生成的那个报文摘要。
5. 对签名进行验证：对分离出的明文邮件 X 进行 MD5 报文摘要运算，得出另一个报文摘要 H（X）。把 H（X） 和前面得出的 H 进行比较，是否和一样。如一样，则对邮件的发送方的鉴别就通过了，报文的完整性也得到肯定。

### 实现效果



## 15. 用实际网络通信实现sm2 2P签名：

### 实现方式



这里面我使用的网络链接代码库为socket

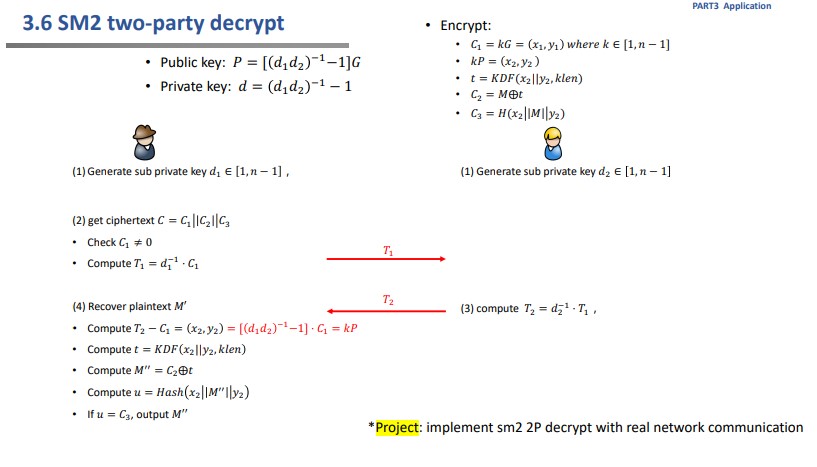
在整个代码的实现中要注意一个事情，就是socket的传送只能传送编码后的字符串，这就意味这在传送点的坐标时在接收方需要将字符串恢复成列表，这样我采用了json.loads（）函数（来自json第三方库）

### 实现效果



## 16. 用实际网络通信实现sm2P解密：

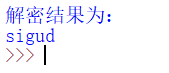
### 实现方式



这由于该项目是解密算法，所以我就提前给A分配d1，B分配d2，并且计算得到协商私钥d，和协商公钥P，然后加密msg “消息摘要”得到密文

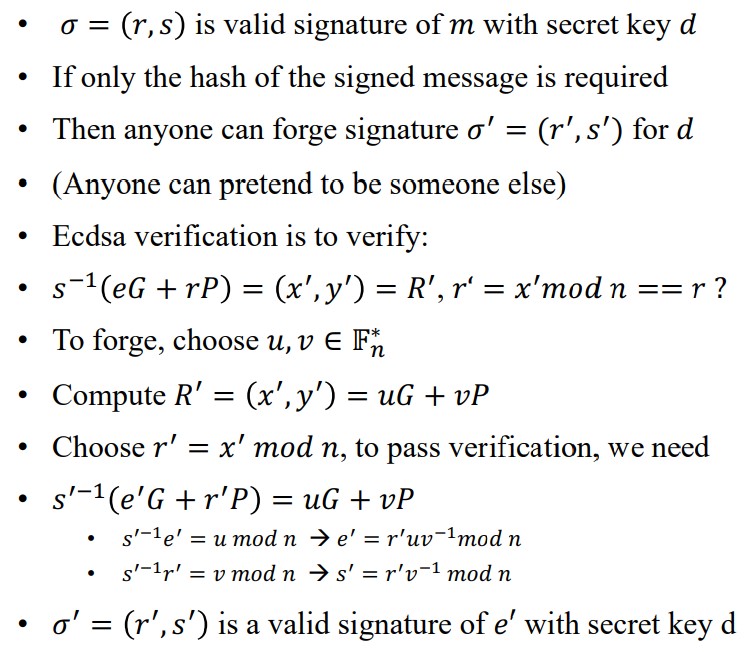
C1， C2， C3 = [（9208123485055232548406805933208065892301341694609167043436851590198577329182， 22497156797805874980417938328502193624670699415281173926964553771202322035334), 428345813838056492794735501498020, 12159561144976371370304274003186071825317752572795393426538011957777595730391]

### 实现效果



## 19. 伪造签名假装你是中本聪：

### 实现方式



### 实现效果

