

# 网络空间安全创新创业实践

## 队员：李君临

## 班级: 网安二班

## 学号：202100460162

## 学院：网络空间安全学院

目录：

[网络空间安全创新创业实践 1](#_Toc142072392)

[队员：李君临 1](#_Toc142072393)

[班级: 网安二班 1](#_Toc142072394)

[学号：202100460162 1](#_Toc142072395)

[学院：网络空间安全学院 1](#_Toc142072396)

[一、组员分工与环境配置 3](#_Toc142072397)

[1、组员分工： 3](#_Toc142072398)

[2、环境配置： 3](#_Toc142072399)

[2.软件环境： 3](#_Toc142072400)

[PyCharm Community Edition 2022.3.3 3](#_Toc142072401)

[Visual Studio 2022 3](#_Toc142072402)

[二、project1: 3](#_Toc142072403)

[1.实验内容及实验思路： 3](#_Toc142072404)

[2.实验结果： 4](#_Toc142072405)

[二、project2： 5](#_Toc142072406)

[1.实验内容及实验思路： 5](#_Toc142072407)

[2.实验结果： 5](#_Toc142072408)

[三、project4： 5](#_Toc142072409)

[1.实验内容及实验思路： 5](#_Toc142072410)

[2.实验结果： 6](#_Toc142072411)

[实验内容及思路： 6](#_Toc142072412)

[五、project9 6](#_Toc142072413)

[1.实验内容及思路： 6](#_Toc142072414)

[2.实验结果： 11](#_Toc142072415)

[六、project14 11](#_Toc142072416)

[1.实验内容及实验思路： 11](#_Toc142072417)

[2.实验结果： 14](#_Toc142072418)

# 一、组员分工与环境配置

## 1、组员分工：

组员：网安二班：李君临

学号：202100460162

任务：完成project1,project2,project4,project8,project9,project14的所有代码编写是最终实验报告撰写。

## 2、环境配置：

1.硬件环境：

处理器 12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz

机带 RAM 16.0 GB (15.7 GB 可用)

系统类型 64 位操作系统, 基于 x64 的处理器

2.软件环境：

PyCharm Community Edition 2022.3.3

Visual Studio 2022

# 二、project1:

## 1.实验内容及实验思路：

实验内容：implement the naïve birthday attack of reduced SM3

实验思路：

1.sm3实现：

（1）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。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

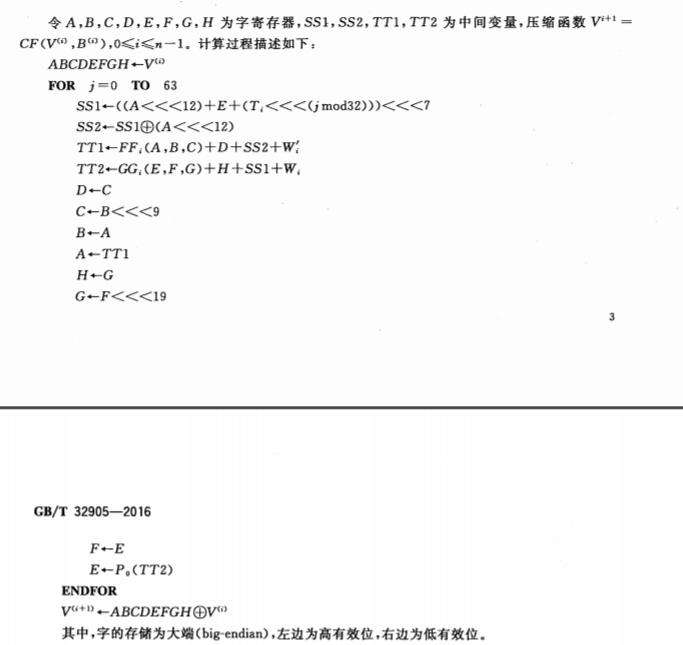
（2）消息扩展

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）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。

（3）迭代压缩

SM3算法迭代压缩过程

初值IV被放在A、B、C、D、E、F、G、H八个32位变量中，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：



（4）输出结果：

将得到的A、B、C、D、E、F、G、H八个变量拼接输出。

（5）生日攻击:

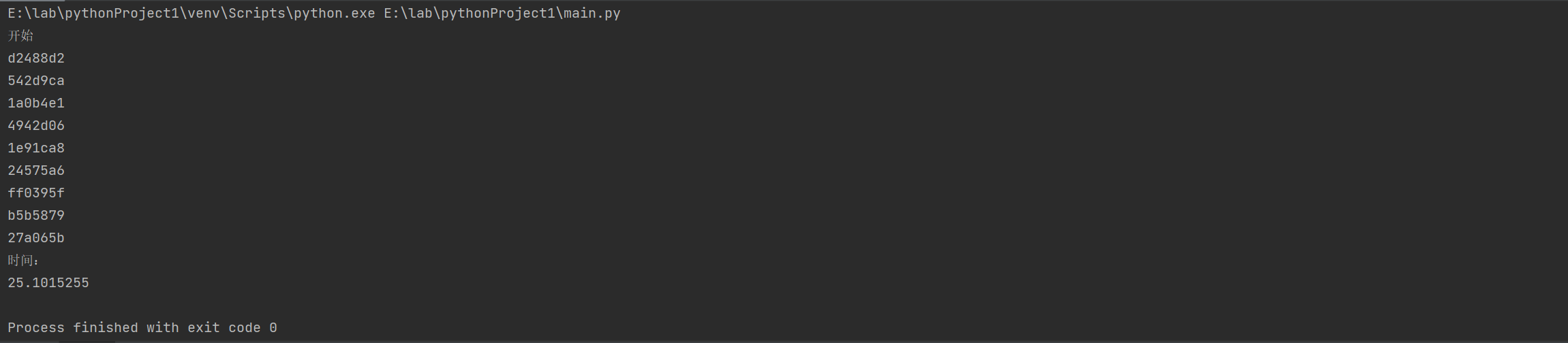
生日攻击指利用哈希函数发生碰撞的可能性，进行n次尝试直到找到一对碰撞的输入。

## 2.实验结果：

|  |  |
| --- | --- |
| SM3长度 | 运行时间 |
| 8bit | 0.0492951s |
| 16bit | 0.8315142s |
| 24bit | 1.4564521s |
| 28bit | 25.111164s |

运行截图：

28bit:



# 二、project2：

## 1.实验内容及实验思路：

实验内容：implement the Rho method of reduced SM3

实验思路：

（1）sm3实现详见project1

（2）rho攻击：

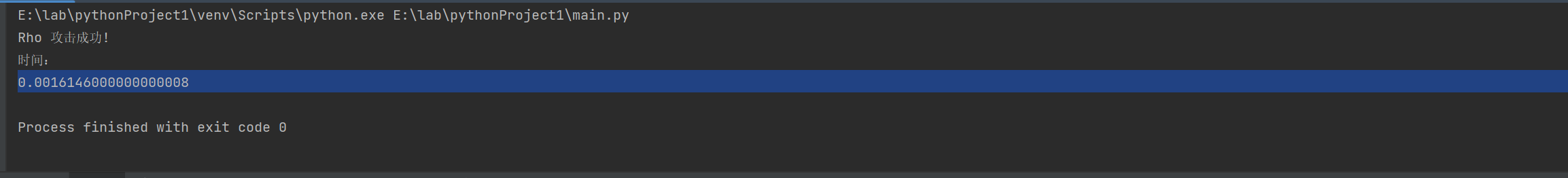
将所有生成的哈希值放入一条链中。

随机生成字符串r，调用自定义的SM3函数计算哈希值，如果当前哈希值在链中，说明此时已经成环，也就意味着找到了碰撞；否则当前哈希值不在链中，就将其添入。

## 2.实验结果：

攻击时间：0.0016146000000000008s

运行截图：



# 三、project4：

## 1.实验内容及实验思路：

实验内容：do your best to optimize SM3 implementation (software)

实验思路：

（1）方案一：用python语言实现sm3

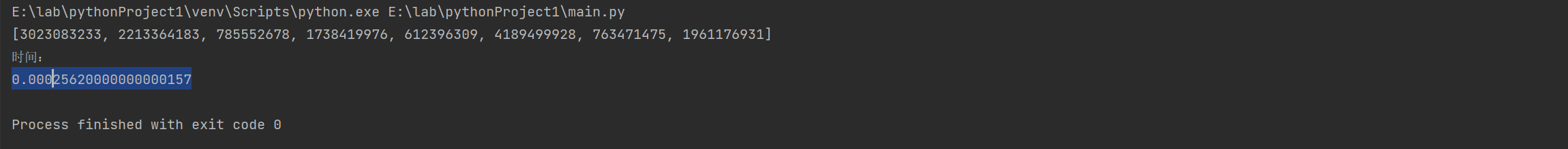
（2）方案二：用c++语言实现sm3

## 2.实验结果：

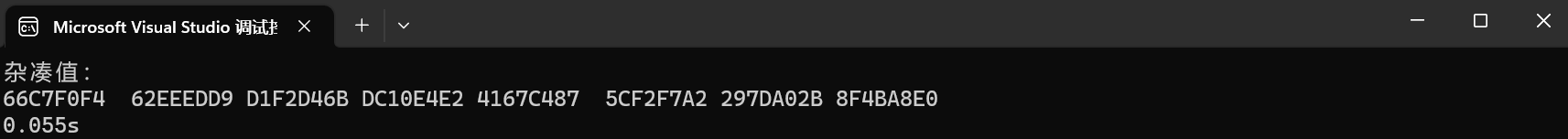
|  |  |
| --- | --- |
| 方案一（python） | 0.00025620000000000157s |
| 方案二（c++） | 0.055s |

运行截图：

方案一：



方案二：

四、project8:

## 实验内容及思路：

实验内容：AES impl with ARM instruction

实验思路：

使用C语言和ARM NEON指令集来编写代码，我们使用ARM NEON指令集中的vaeseq\_u8函数来执行AES加密。该函数执行16字节的AES加密操作，将明文数据和密钥数据作为输入，然后输出加密后的数据。

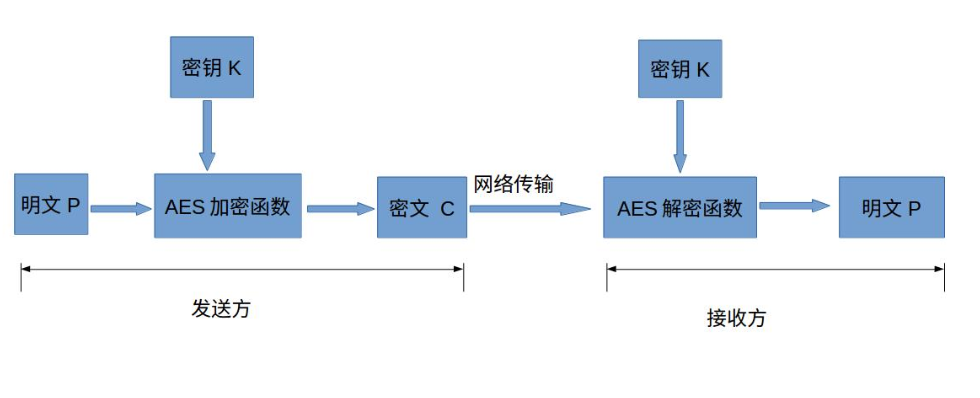
# 五、project9

## 1.实验内容及思路：

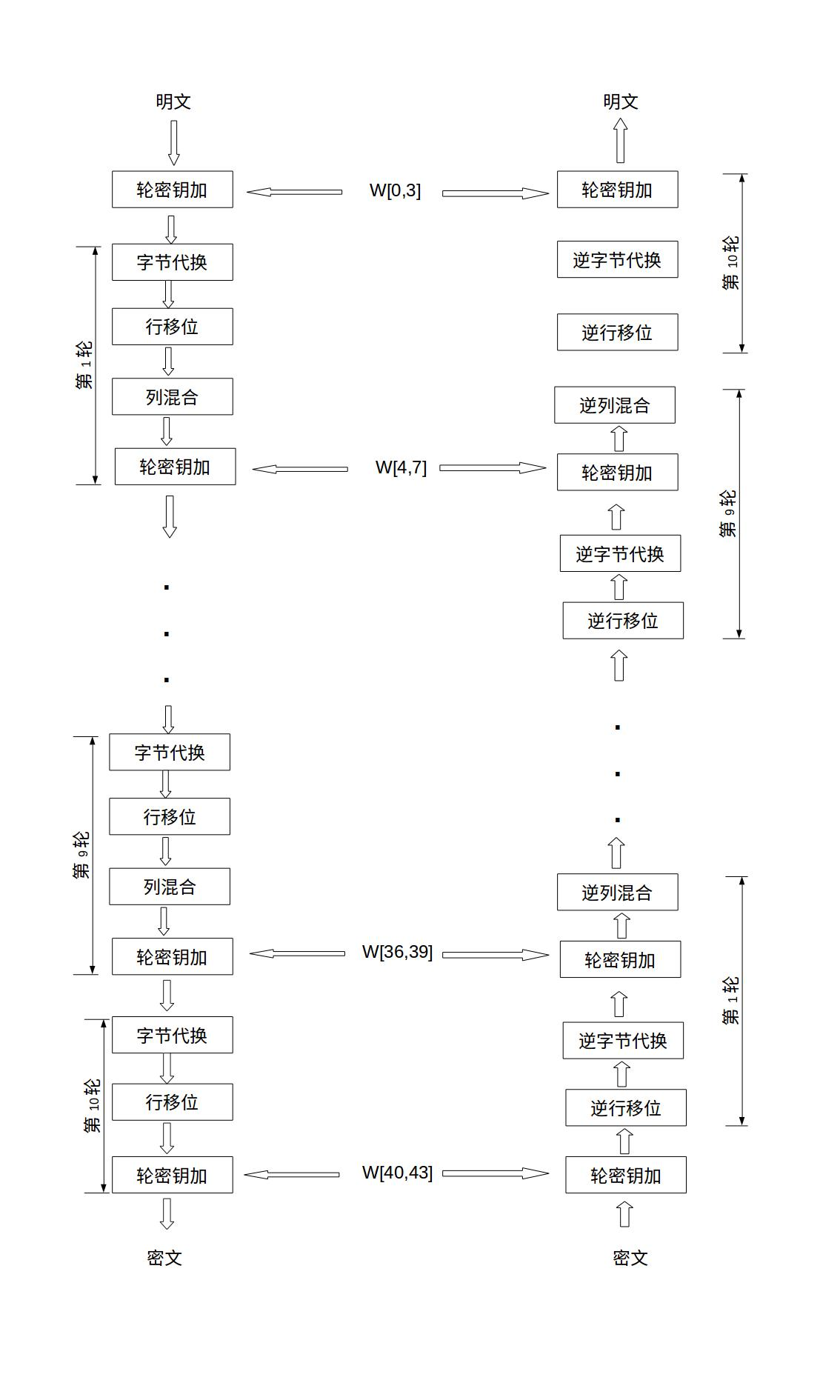
实验内容：AES实现

实验思路：

AES流程：



AES的基本结构：



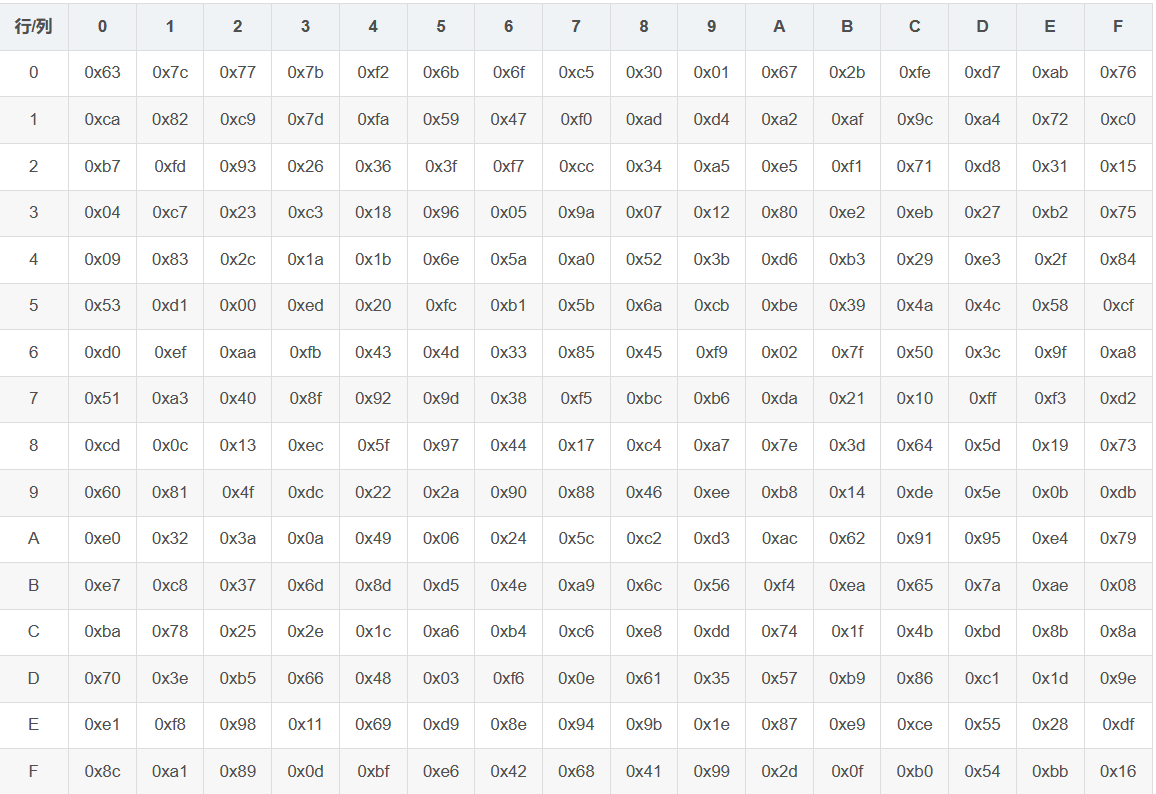
每一步的具体实现：

一、字节代换

字节代换操作

AES的字节代换其实就是一个简单的查表操作。

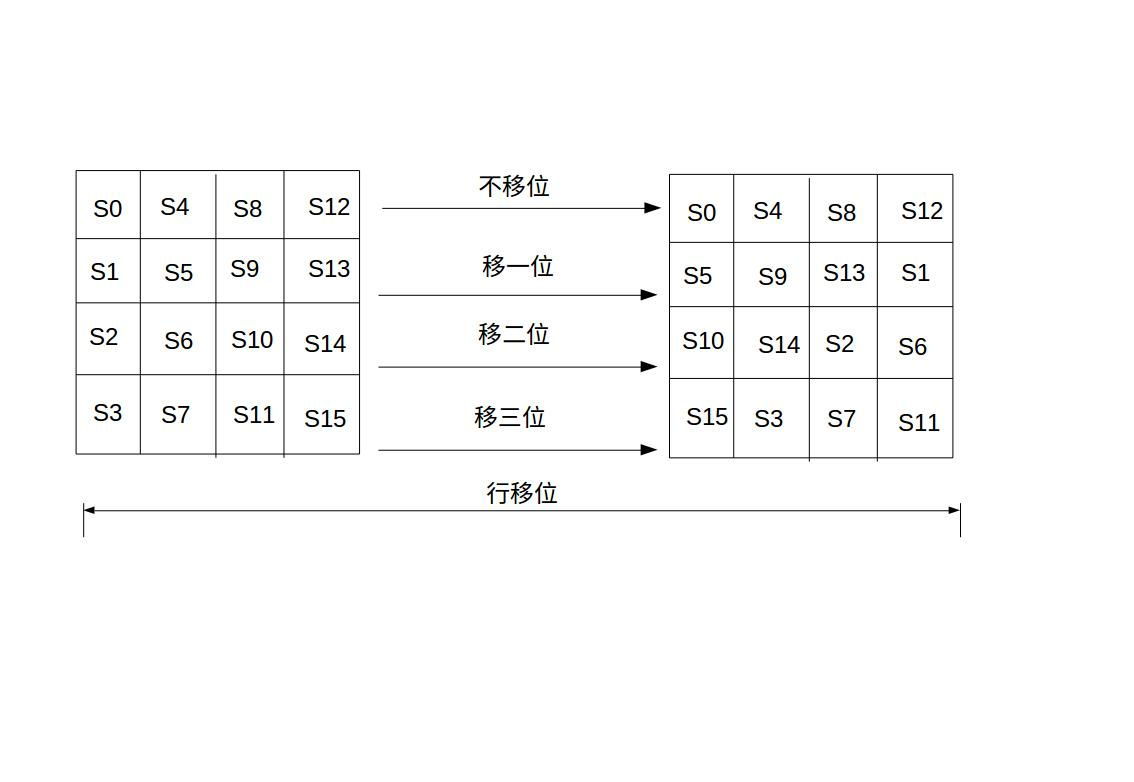
AES的S盒：

状态矩阵中的元素按照下面的方式映射为一个新的字节：把该字节的高4位作为行值，低4位作为列值，取出S盒中对应的行的元素作为输出。

二、行移位

行移位操作

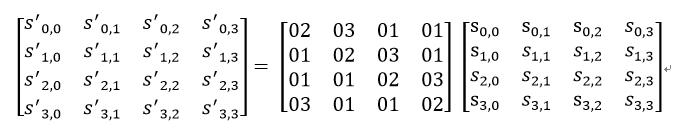
行移位是一个简单的左循环移位操作。当密钥长度为128比特时，状态矩阵的第0行左移0字节，第1行左移1字节，第2行左移2字节，第3行左移3字节，如下图所示：



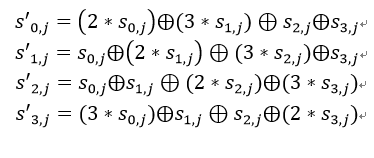
三、列混合

列混合操作

列混合变换是通过矩阵相乘来实现的，经行移位后的状态矩阵与固定的矩阵相乘，得到混淆后的状态矩阵，如下图的公式所示：

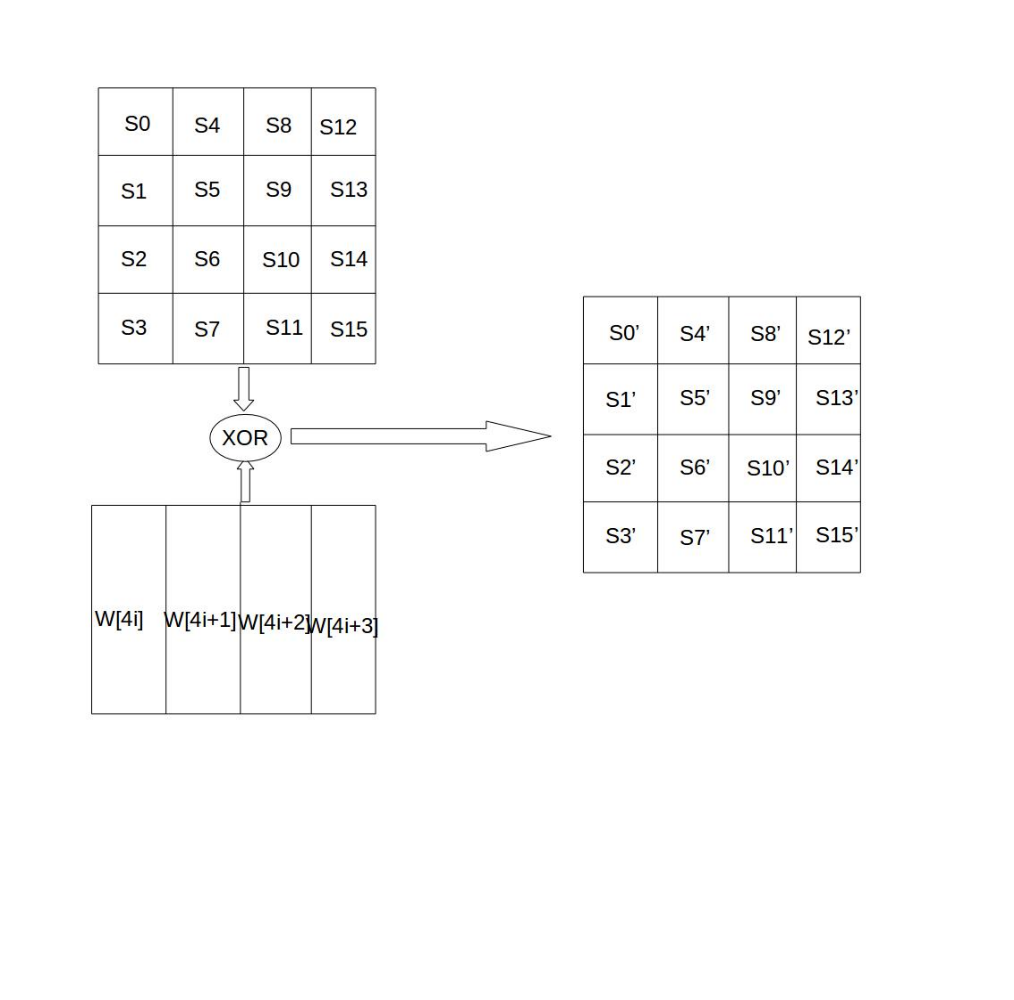


状态矩阵中的第j列(0 ≤j≤3)的列混合可以表示为下图所示：



四、轮密钥加

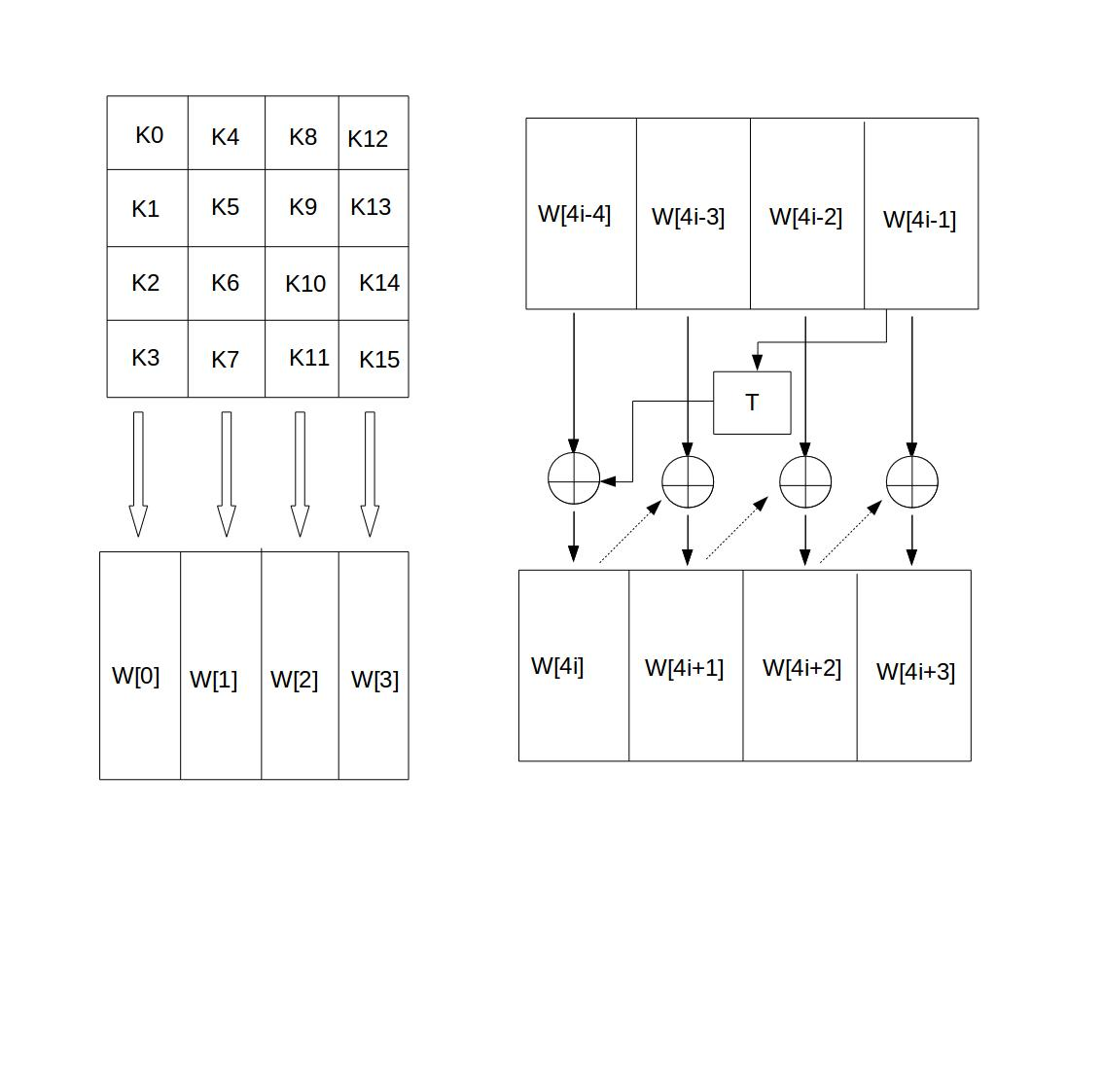
轮密钥加是将128位轮密钥Ki同状态矩阵中的数据进行逐位异或操作。



其中，密钥Ki中每个字W[4i],W[4i+1],W[4i+2],W[4i+3]为32位比特字，包含4个字节，他们的生成算法下面在下面介绍。轮密钥加过程可以看成是字逐位异或的结果。

密钥扩展：

AES首先将初始密钥输入到一个44的状态矩阵中，如下图所示。



这个44矩阵的每一列的4个字节组成一个字，矩阵4列的4个字依次命名为W[0]、W[1]、W[2]和W[3]，它们构成一个以字为单位的数组W。

接着，对W数组扩充40个新列，构成总共44列的扩展密钥数组。新列以如下的递归方式产生：

1.如果i不是4的倍数，那么第i列由如下等式确定：

W[i]=W[i-4]⨁W[i-1]

2.如果i是4的倍数，那么第i列由如下等式确定：

W[i]=W[i-4]⨁T(W[i-1])

其中，T是一个有点复杂的函数。

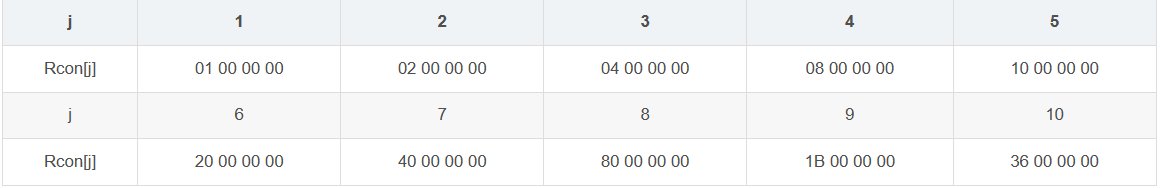
函数T由3部分组成：字循环、字节代换和轮常量异或，这3部分的作用分别如下。

a.字循环：将1个字中的4个字节循环左移1个字节。即将输入字[b0, b1, b2, b3]变换成[b1,b2,b3,b0]。

b.字节代换：对字循环的结果使用S盒进行字节代换。

c.轮常量异或：将前两步的结果同轮常量Rcon[j]进行异或，其中j表示轮数。

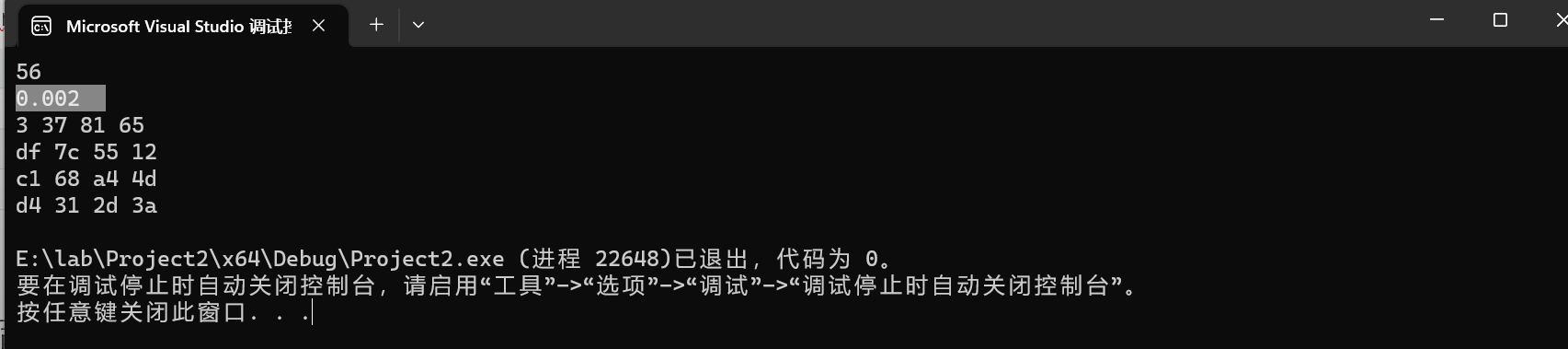
轮常量Rcon[j]是一个字，其值见下表。



## 2.实验结果：

时间：0.002

运行截图：



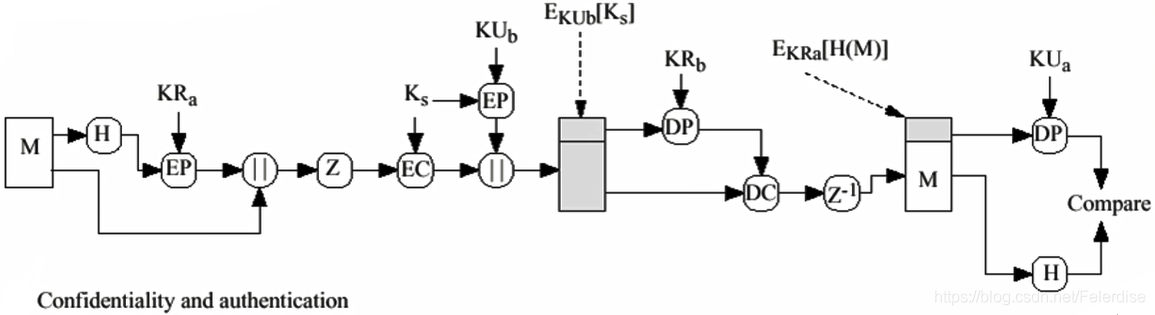
# 六、project14

## 1.实验内容及实验思路：

实验内容: Implement a PGP scheme with SM2

实验思路：

PGP加密流程：

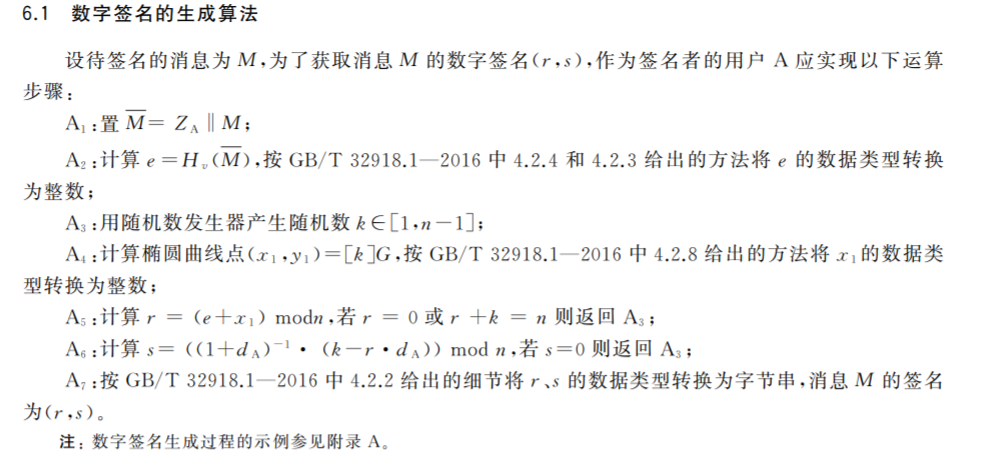


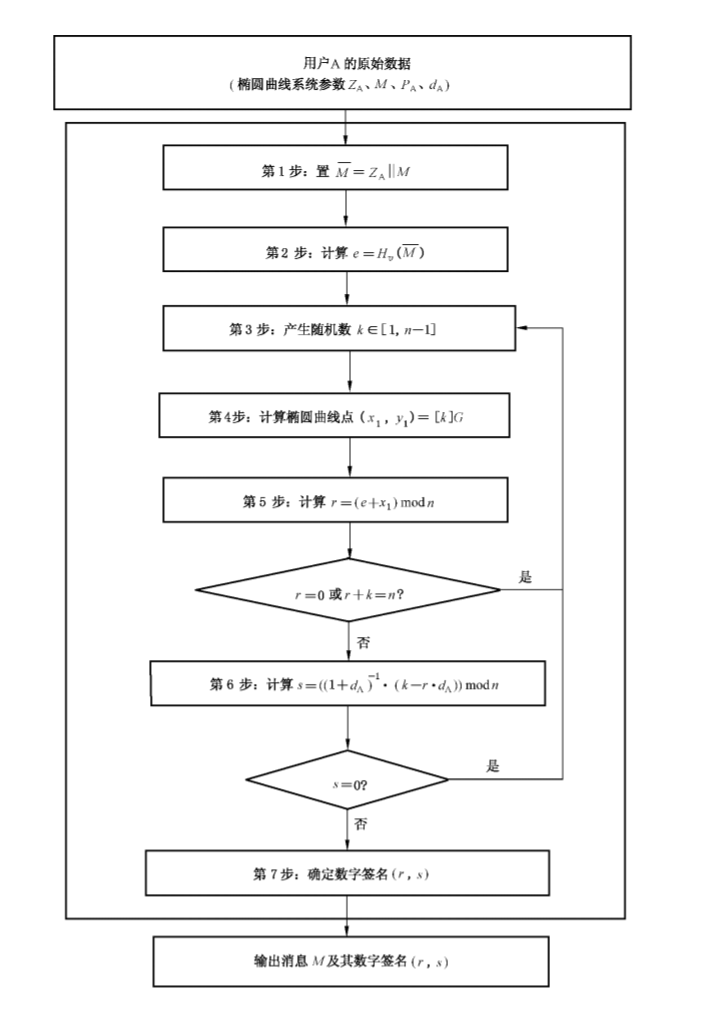
假设在一个数据M传输的过程中，发送者为A、接受者为B，公私钥对( p k , r k )，会话密钥s k。

其中哈希函数为sm3，公钥密码算法为sm2，对称加密算法为sm4。

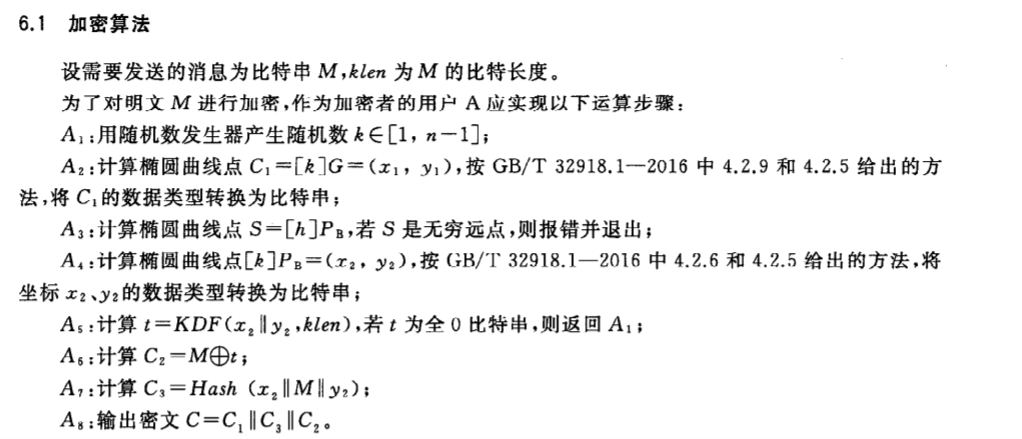
实现结果：**SM2pkB​​(sk)∣∣SM4sk​(sigSM2−rkA​​(SM3(M))∣∣M)**

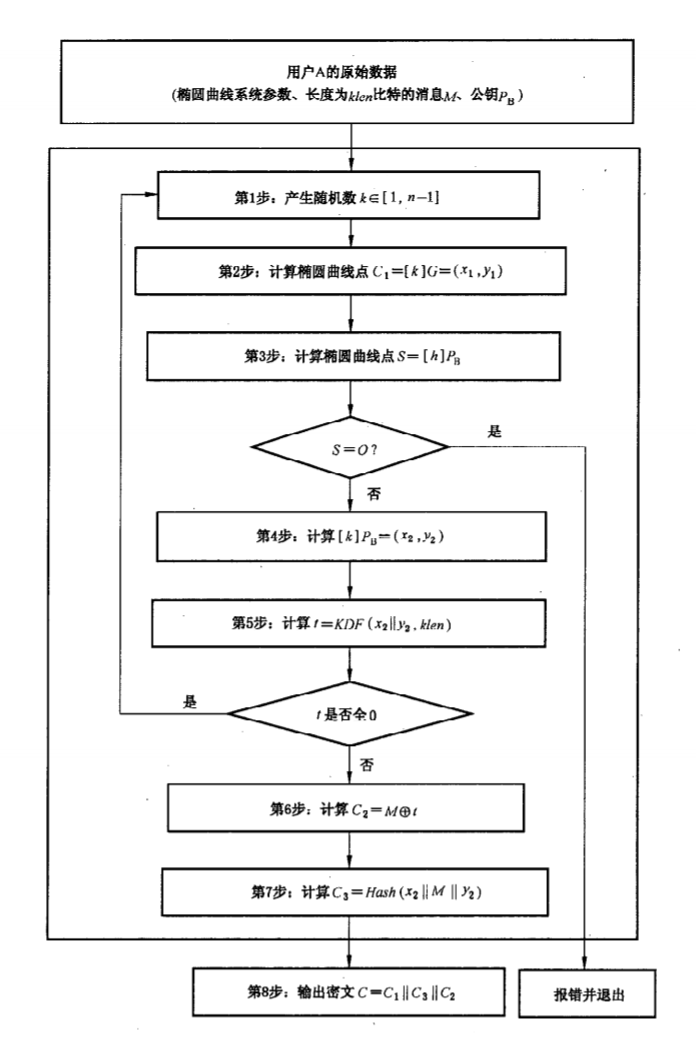
Sm2签名实现：





Sm2加密实现：

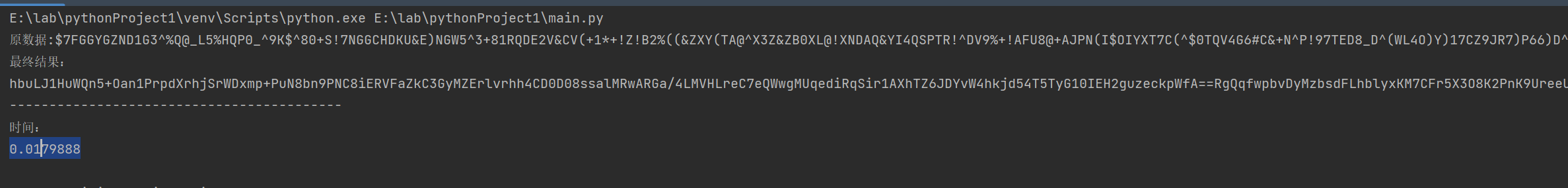




## 2.实验结果：

时间：0.0179888

运行截图：



# 七、实验总结：

Project1: 实现简化版SM3的朴素生日攻击 在此项目中，我们将实现SM3哈希算法的简化版本，并进行朴素生日攻击。生日攻击是一种哈希碰撞攻击，通过寻找两个不同的输入，它们的哈希值相同。我们将实现这个攻击并测试其效果。

Project2: 实现SM3的Rho方法 Rho方法是一种巧妙的SM3哈希碰撞攻击技术，它利用了循环节和概率统计的方法来寻找碰撞。我们将实现Rho方法，并与朴素生日攻击进行比较，验证其性能和效果。

Project8: 使用ARM指令集实现AES加密 在这个项目中，我们将使用ARM指令集的AES扩展来实现AES对称加密算法。通过利用硬件加速的AES指令，我们将提高加密速度，并与软件实现进行对比。

Project9: 实现AES软件加密 在此项目中，我们将实现AES密码算法的软件加密。通过这个实验，我们将深入了解这个算法的细节，并测试它们的加密性能。

Project4: 优化SM3软件实现 在这个项目中，我们将对SM3哈希算法的软件实现进行优化。通过采用更高效的算法、数据结构和优化技术，我们将提高SM3的运行速度。

Project14: 使用SM2实现PGP加密方案 在这个项目中，我们将实现基于SM2椭圆曲线加密算法的PGP（Pretty Good Privacy）加密方案。PGP是一种广泛使用的加密方案，结合SM2算法将为我们提供更高的安全性。

通过以上实验，我们深入了解了SM3、AES、SM4、SM2等密码学算法的原理和实现细节。我们对哈希碰撞攻击、硬件加速、优化技术等内容有了更深入的了解。通过实践，我们进一步巩固了密码学的基本概念，并提升了密码学算法的编程和实现能力。同时，我们意识到在实际应用中，密钥管理、随机数生成等问题也是至关重要的，需要在实际应用中加以考虑和实现。总体而言，本实验为我们进一步学习和应用密码学奠定了坚实基础。