

**网络空间安全创新创业实践project报告**

|  |  |  |
| --- | --- | --- |
| 姓名 | ： | 王成盟 |
| 专业 | ： | 网络空间安全 |
| 班级 | ： | 21级2班 |
| 学号 | ： | 202100460091 |

**目 录**

[Project1 1](#_Toc17198)

[说明： 1](#_Toc26755)

[结果： 1](#_Toc25832)

[Project2 1](#_Toc32379)

[说明： 1](#_Toc18856)

[结果： 2](#_Toc5088)

[Project3 3](#_Toc6063)

[说明： 3](#_Toc29644)

[结果： 4](#_Toc15365)

[Project4 4](#_Toc21534)

[说明： 4](#_Toc26167)

[结果： 5](#_Toc23211)

[Project5 5](#_Toc3908)

[说明： 5](#_Toc17677)

[结果： 6](#_Toc19621)

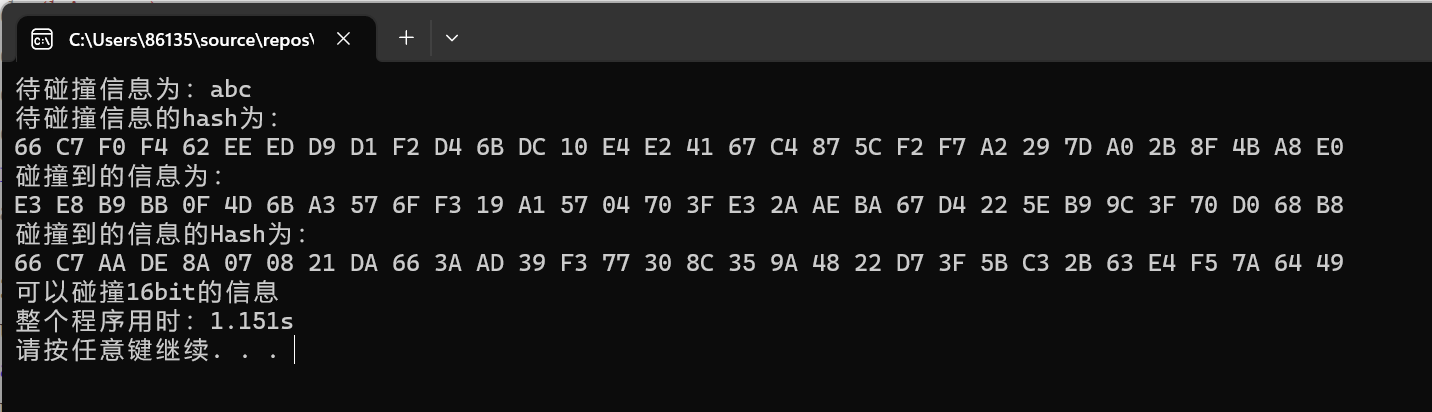
处理器：AMD Ryzen 5 5600H with Radeon Graphics 3.30 GHz

# Project1

## 说明：

按照rho方法，将输出作为下一步的输入，之后迭代循环，直到hash值发生碰撞，因sm3的碰撞开销很大，所以将长度缩短减少时间。

## 结果：



用时1.151s

# Project2

## 说明：

在哈希长度扩展攻击中，恶意攻击者利用哈希函数的特性。在已知消息M的哈希值H(M)和消息M的情况下，攻击者构造另一消息M'，以便可以计算出M'的哈希值H(M')，而无需了解M'的确切内容。SM3哈希算法的长度扩展攻击利用了Merkle-Damgård结构的特性。

伪代码：

Initialize IV (Initialization Vector) with some constants

Divide the input message M into blocks: M = M1 || M2 || ... || Mn

for each block Mi:

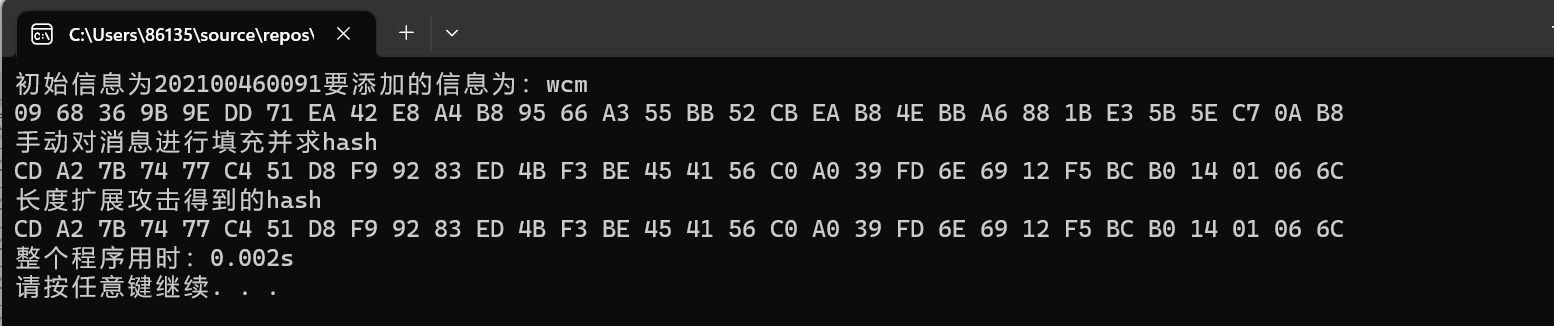
Compress(IV, Mi) # Compress the IV and the block Mi to update IV

Output the final value of IV as the hash value H(M)

压缩函数被称为Compress函数。它接受当前的初始向量IV和消息块Mi作为输入，并生成新的IV作为输出。在长度扩展攻击中，攻击者已知消息M的哈希值H(M)，并试图构建新的消息M'，使得可以计算出H(M')。

在构建新消息块M'时，必须根据原始消息的长度进行适当的数据填充，以确保新消息的长度满足哈希函数的块长度要求。

## 结果：



用时：0.002s

# Project3

## 说明：

SM3适合用于商业密码应用中的数字签名和验证，是一种在基于SHA-256的基础上改进实现的算法，其安全性与SHA-256相当。SM3与MD5的迭代过程相似，同样采用Merkle-Damgard结构。消息分组长度为512位，摘要值长度为256位。

整个算法的执行过程可归纳为四个主要步骤：消息填充、消息扩展、迭代压缩和输出结果。

由于其Merkle-Damgard结构，SM3无法通过多线程同时加密消息分组，因为后续结果将依赖前面的结果。因此，采用预计算64个常量、多线程多次加密不同文件，并在关键步骤中展开循环，然后同时调用多个SM3模块以进行多线程优化，以提高吞吐量。

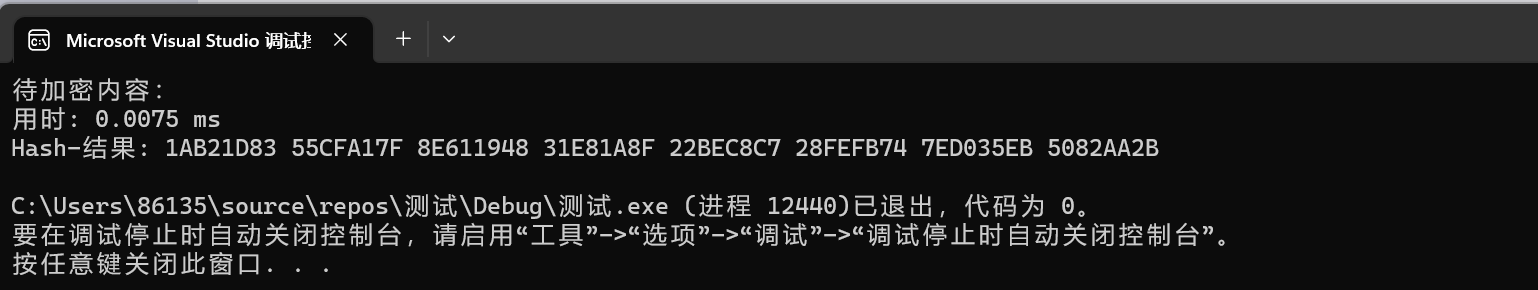
现代CPU采用流水线设计，将一条机器指令的执行分成多个阶段，每个时钟周期尽可能多地执行不同指令的不同阶段，这就是流水线。为了优化流水线，可以通过循环展开来增加每次迭代计算的元素数量，从而减少循环迭代次数，提高指令并行性，并减少数据相关性，以增强流水线效果，就像汽车生产流水线一样。

采用多个线程，每个线程完成多次SM3计算，计算多次加密不同文件，然后同时调用多个SM3模块进行多线程优化，可以有效提高吞吐量。

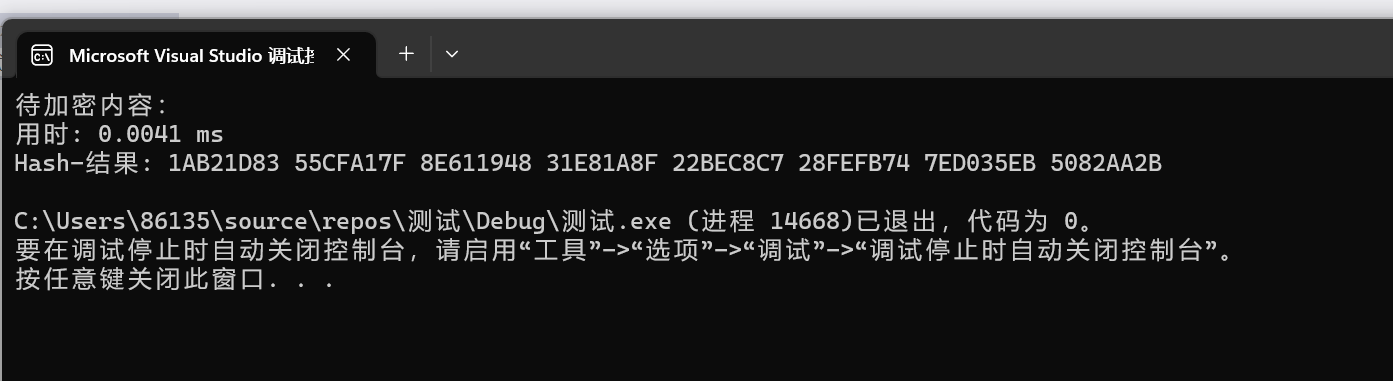
为了优化，可以预先计算并存储64个常量。尽管这会牺牲256字节的存储空间，但可以避免每个消息分组都执行常数移位操作。此外，运算时间几乎与消息大小成正比，消息越大，优化效果越明显，从而大幅缩短运算时间，提高性价比。

## 结果：

未优化：



优化后：



优化前用时0.0075ms，优化后用时0.0041ms

# Project4

## 说明：

在选择AES的密钥长度方面，可从128位、192位和256位这三个选项中挑选合适的密钥长度，以根据实际需要进行选择。

基于所选的密钥长度，需进行密钥扩展以生成轮密钥，用于后续的加密过程。

对待加密的数据进行分组时，按照每组128位（16字节）来进行分割。

初始数据组与密钥的第一组进行异或运算。

在进行迭代加密时，每一轮包含以下操作：

字节代换：使用S-Box（Substitution Box）替换数据块中的每个字节。

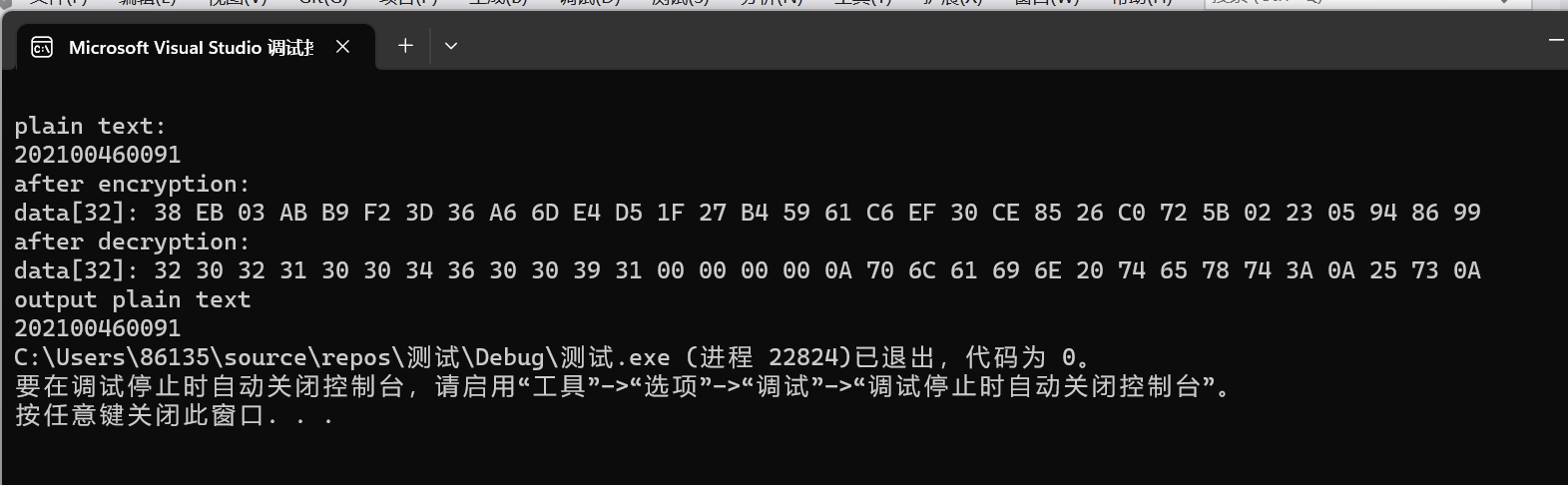
行移位：对数据块的每一行进行循环左移操作。

列混淆：通过矩阵乘法对数据块进行列混淆。

轮密钥加：将当前轮的轮密钥与数据块进行异或运算。

最后一轮不包含列混淆步骤，完成最后一轮迭代加密。

## 结果：



# Project5

## 说明：

MPT 的结构如下：

每个节点包含一个或多个 Key-Value 对。

Key 是数据的哈希值或一系列字节组成的前缀，用于唯一标识一个 Value。

Value 可以是实际数据，也可以是另一个 MPT 节点的哈希值，从而形成递归结构。

prefix就是HP编码！对终止符的状态和奇偶性进行编码。最低位表示奇偶性，第二低位编码终止符状态。

总共有2个扩展节点，2个分支节点，4个叶子节点。 右边是叶子节点的情况，左边代表的是节点的prefix（HP编码）

这就是一个状态树的存储形式，其实他应当长的样子，我们可以细细想一下，他的key被编码成一种特殊的16进制的表示，value是一些rlp后的数据，而且比上图要大的多的多。

插入操作基于查找完成，首先找到与新插入节点拥有最长相同路径前缀的节点，之后基于此节点将剩下的节点进行封装插入。

## 结果：

![`KZ)@X2I0KP2](@[W[%LH%4](data:image/png;base64,)