**第一次编程作业实验报告**

刘曼姝 1901210656 2019.10.15

1. **实验目的与要求：**

实现对SM3哈希函数做长度扩展攻击的可展示实例。

1. **实验设备、系统环境与软件：**

Windows 10 64位操作系统的笔记本电脑，安装了ActivePerl 5.20.2、Visual Studio 2019、GmSSL 2.5.4、Notepad++。

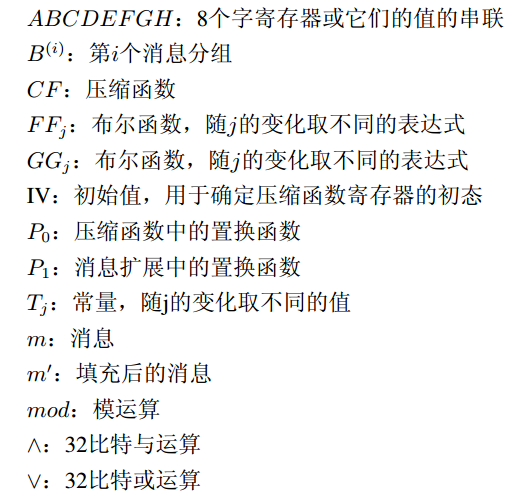
1. **实验原理：**

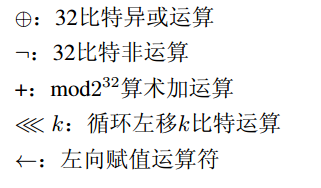
**1、SM3简介**

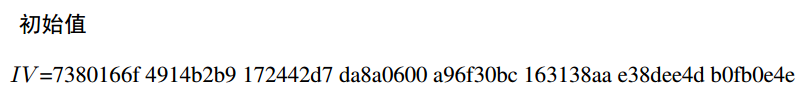
SM3是国产的哈希函数（标准），在商用密码体系中主要用于数字签名及验证、消息认证码生成及验证、随机数生成等，其算法公开[1]，在2010年作为国家加密标准发布，使用于敏感但非机密的应用中。SM3生成256位哈希值，使用128位分组和256位椭圆曲线密码算法（公钥密码算法）。

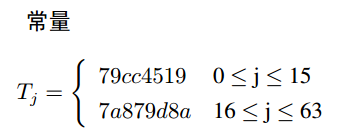
**2、SM3算法[2]**

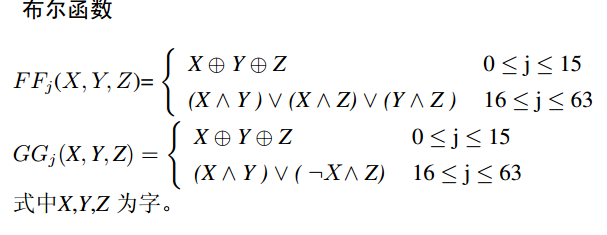
**（0）SM3算法中运用的符号、常量与函数定义：**

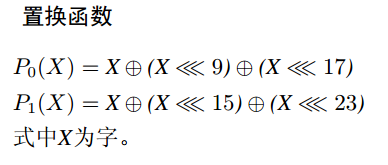










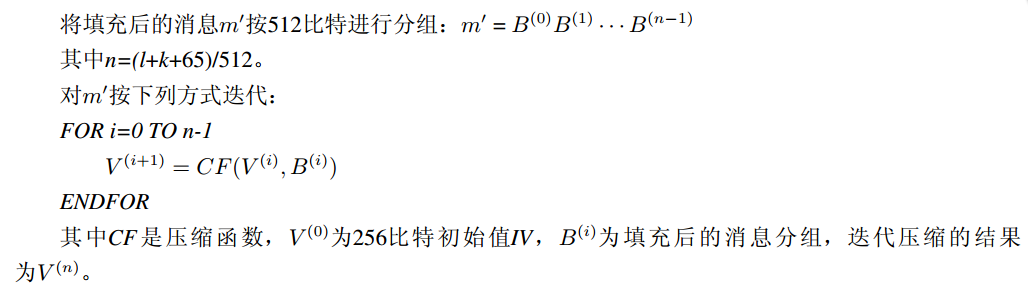


**（1）填充：**

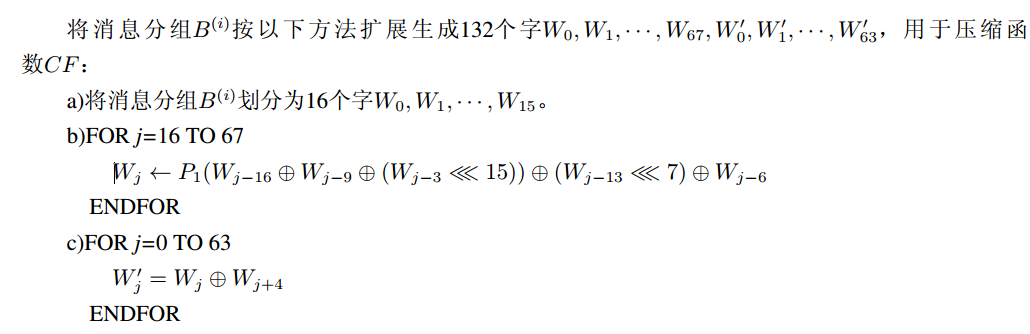
假设消息m长度为Lbit。首先将1bit的“1”添加到消息末尾，然后再添加k个“0”，k是满足L+1+k≡448(mod 512)的最小非负整数，最后再添加64bit用来表示长度L（二进制），填充后的消息m′的比特长度即为512的倍数。

**（2）迭代压缩：**

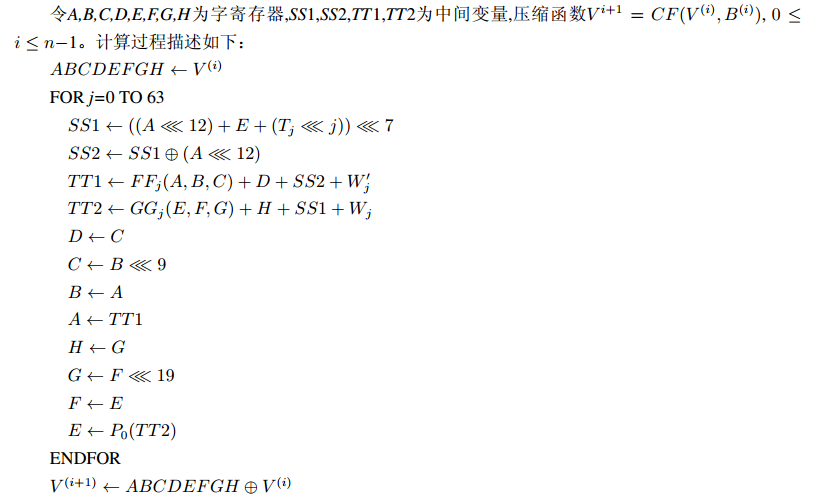
① 迭代：



② 消息扩展：

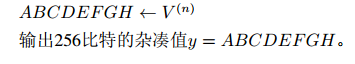


③ 压缩：



其中，字的存储为大端格式。

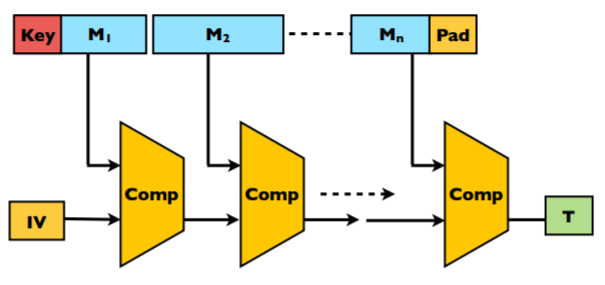
④ 杂凑值：



**3、长度扩展攻击[3]**

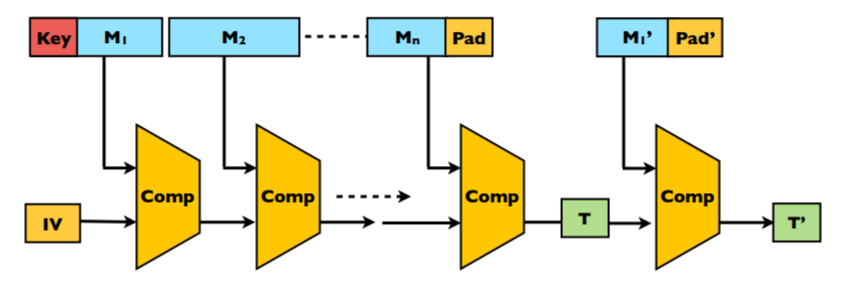
长度扩展攻击（length extension attack）指针对某些允许包含额外信息的加密哈希函数的攻击手段，攻击者可以在不知道密钥K的情况下，附加任何数据并生成有效的哈希。Merkle–Damgård形式的哈希函数易受长度扩展攻击，SM3就属于这种形式。

Merkle–Damgård形式的哈希函数原理如图，可表示成T=H(K || M)形式：



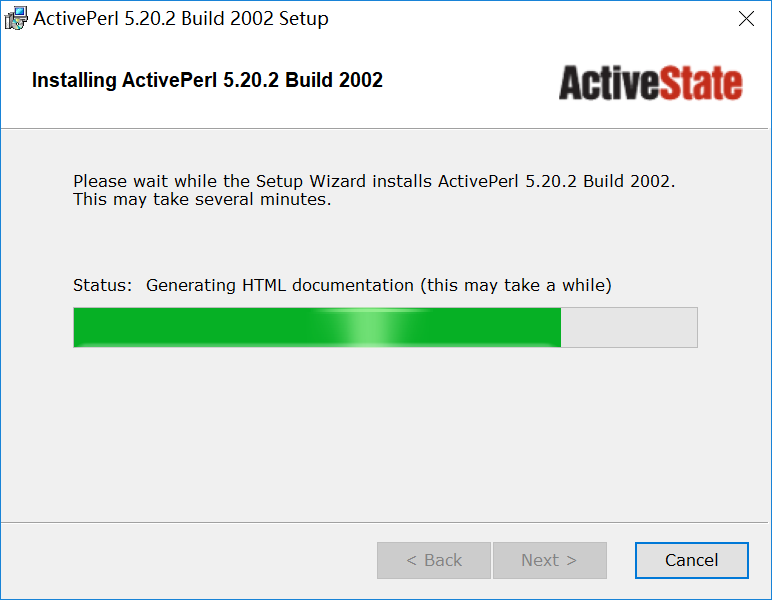
对于T=H(K || M)形式的哈希，在以下条件满足的情况下，攻击者可以通过该方法获取“H(K || 一定规则构造的M)”：

1. 知道H的算法且该算法满足Merkle–Damgård哈希函数特征；
2. 不知道或不可控K的具体值，但知道K的长度，并可控制M的值；
3. 可以得到T=H(K || M)的值。



即可根据上图，通过已知的K的长度，确定Pad填充，并将扩展的消息M’附加在后面，形成T’ = Hash(K || M || Pad || M’)。而在攻击的实际操作中，将对M’操作的这一轮的初始向量IV改为H(K || M)，即可得到与T’ = Hash(K || M || Pad || M’)相等的哈希值，从而伪造有效的哈希值。

1. **实验步骤与设计思路：**
2. **GmSSL在Window下的编译和安装[4]**
3. **安装ActivePerl（Visual Studio之前已安装）：**



（安装好后，如果没有勾选设置路径，可以输入命令：

set path=C:\Users\0\Downloads\ActivePerl\perl\bin;%path%;

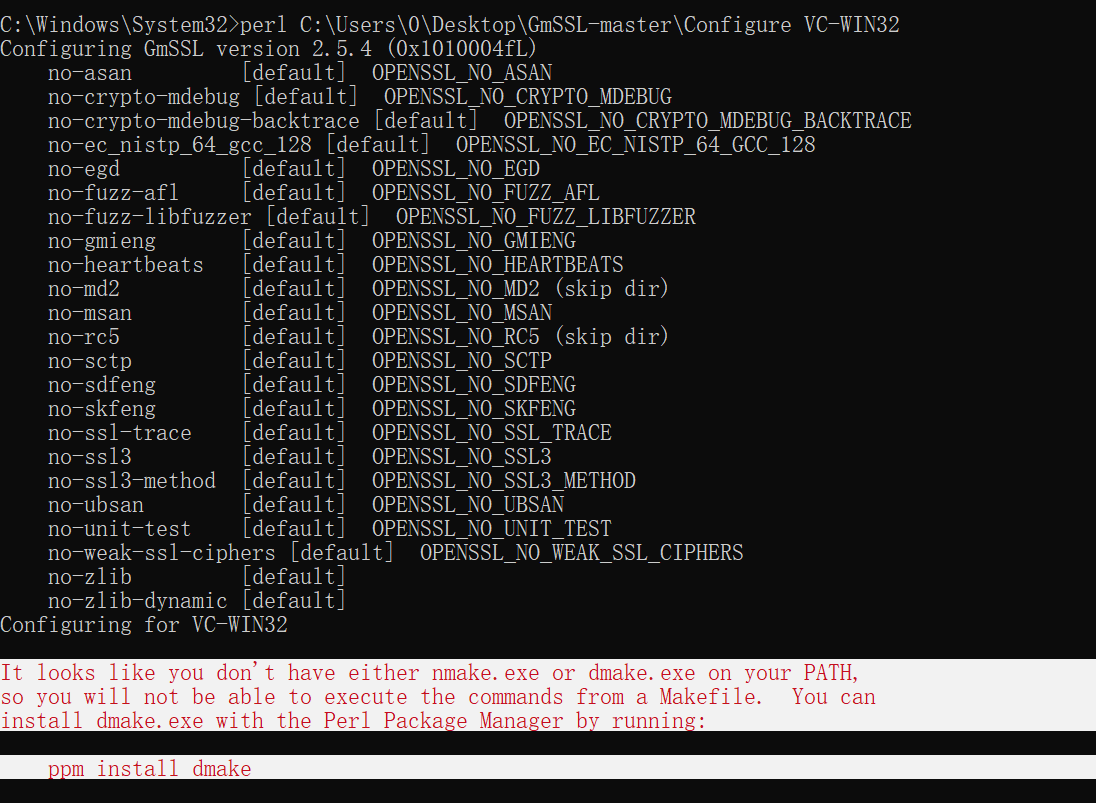
设置perl路径；输入命令：

perl -v

查看是否设置正确。）

**（2）以管理员身份打开Visual Studio Tools下的Developer Command Prompt控制台并运行命令：**

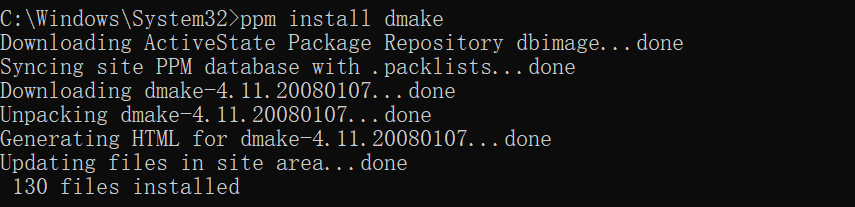
perl C:\Users\0\Desktop\GmSSL-master\Configure VC-WIN32





提示PATH中没有nmake.exe、dmake.exe，需要安装，输入命令：

ppm install dmake

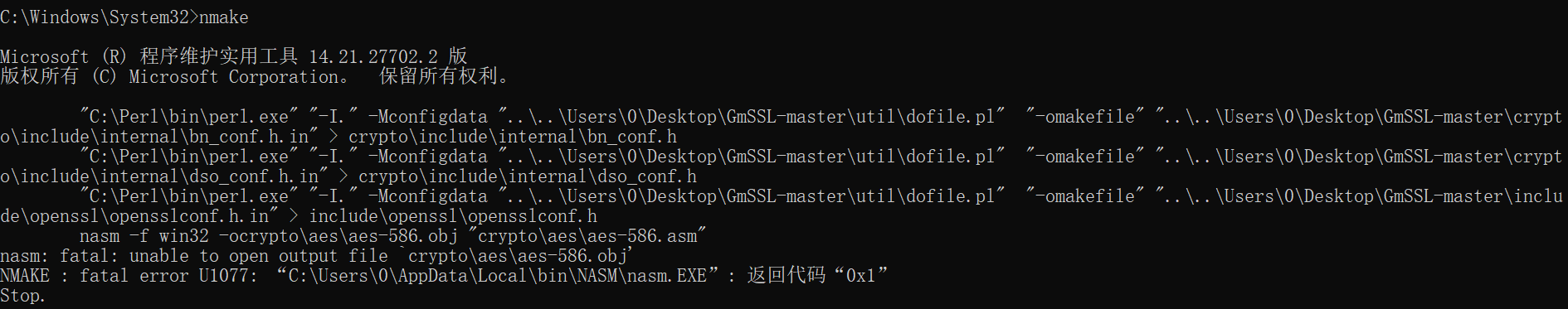


然后输入命令：nmake

此时提示错误：

nasm: fatal: unable to open output file `crypto\aes\aes-586.obj'

NMAKE : fatal error U1077: “C:\Users\0\AppData\Local\bin\NASM\nasm.EXE”: 返回代码“0x1”

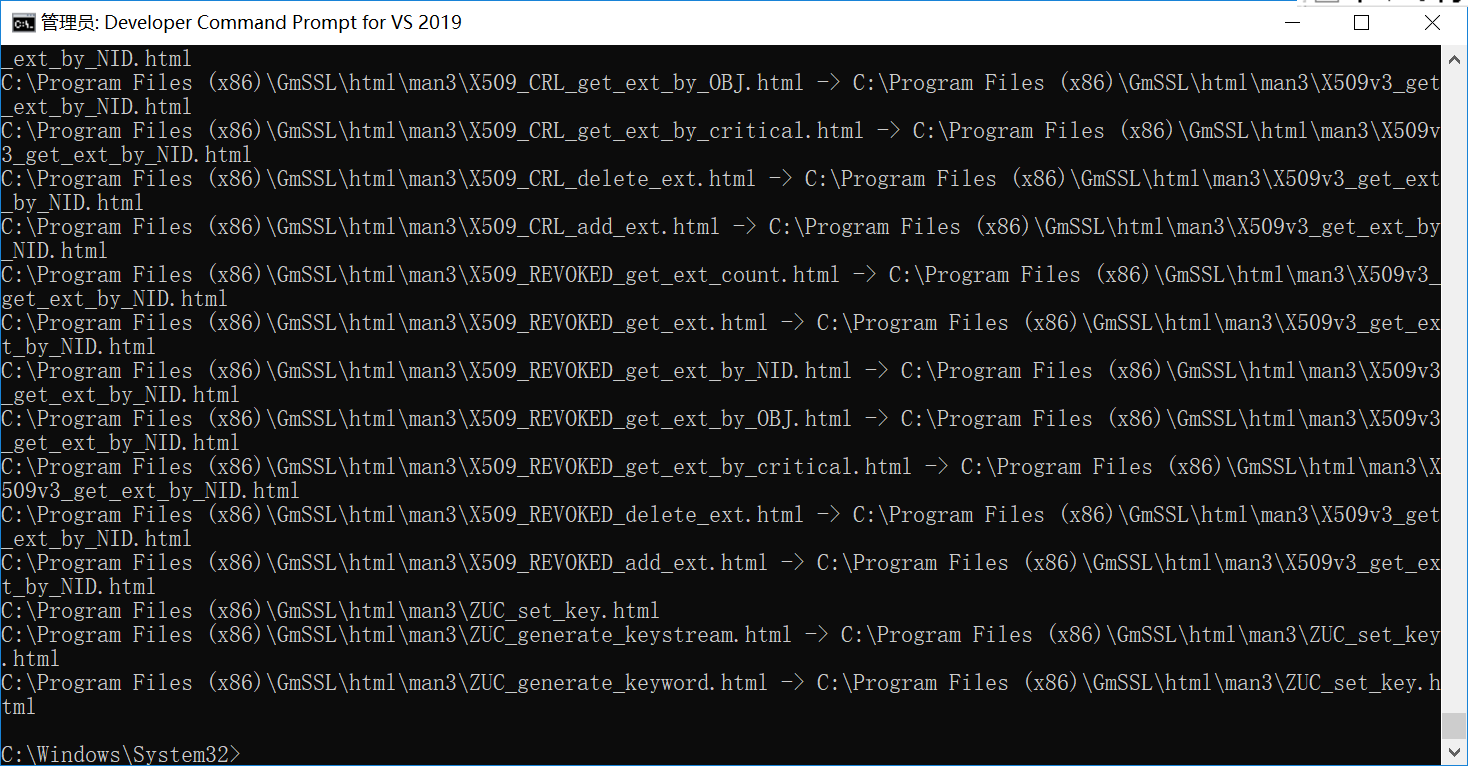


解决方法是把上一步命令换成：

perl C:\Users\0\Desktop\GmSSL-master\Configure VC-WIN32 no-asm

再次Configure，然后再输入命令：nmake。

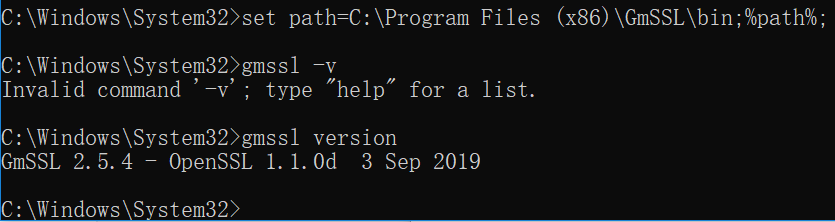
最后输入命令：nmake install



安装之后执行gmssl命令行工具检查是否成功：

set path=C:\Program Files (x86)\GmSSL\bin;%path%;

gmssl version



说明安装成功。

**2、SM3的长度扩展攻击[5]**

**（1）获得原有“密钥 || 消息”的SM3哈希值T=H（K || M）：**

根据T = Hash(K || M)，设密钥K=“Passw0rd”，长度为8；消息M=“abc”，长度为3；将K || M存储在sm3.txt中，其总长度为11。

输入命令：

gmssl sm3 C:\Users\0\Desktop\sm3.txt

得到哈希值为：

“e0108e33795dc84661bcb9afcd3bf5d62cce9c97cb397f43073fc07afee23387”。

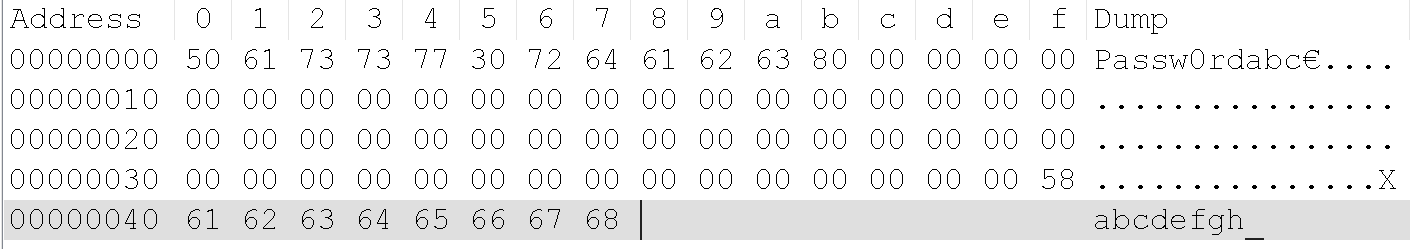


1. **求长度扩展攻击后应得的SM3哈希值T’ = Hash(K || M || Pad || M’)：**

（首先可以在Notepad++中安装插件HexEditor.dll，以方便打16进制数。）

以T’ = Hash(K || M || Pad || M’)的形式做长度扩展。因为K || M的总长度为11个字节，即11\*8=88=\x58bit，填充的最后还要预留8个字节的长度字段（SM3采用大端存储），故不表示长度的填充字段应为64-11-8=45字节，因此：

Pad=“\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x58”； 设在此后填充的M’=“abcdefgh”，然后将K || M连接在前面，形成“K || M || Pad || M’”：





存储在sm31.txt中。输入命令：

gmssl sm3 C:\Users\0\Desktop\sm31.txt

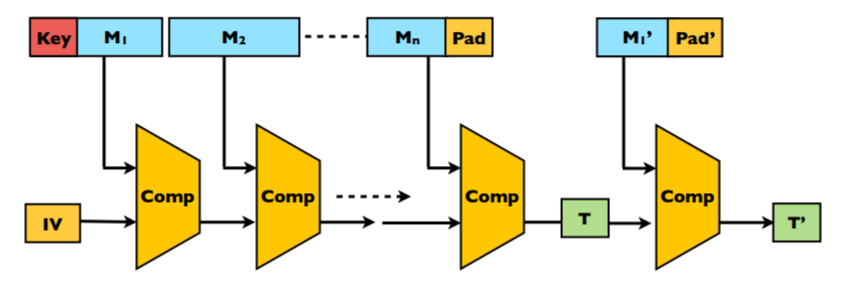
得到长度扩展后的哈希值应该为：

“aa84972131db07e8674a11ba46d373bce433b0f510d996808d314bf6e894dacd”。



服务器将根据该值检查发送的数据。

1. **执行实际的攻击：**

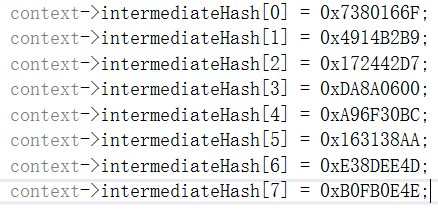
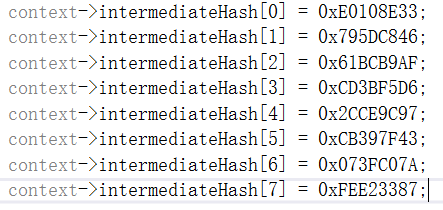


根据该原理图，思路是求：

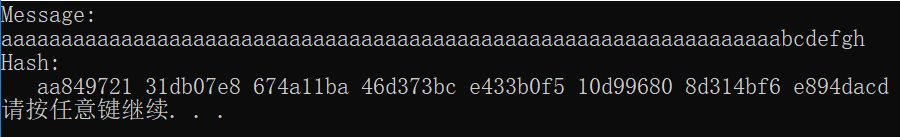
“aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabcdefgh”的哈希值，其中前64个字符可以是任意的，只要其长度是64，能让Hash函数做一块运算即可。

修改程序，使得对如上字符串的前64个字符（一个整块）还以原来的初始向量IV做运算，而对于接下来超过64个字符的M’部分，则将初始向量设置为H(K || M)：

“e0108e33795dc84661bcb9afcd3bf5d62cce9c97cb397f43073fc07afee23387”，以4个字节为一组：

改为。

运行程序得到如下实验最终结果：



**五、实验结果分析：**

从实验结果可以看出，攻击者使用“任意64字节 || M’=abcdefgh”计算出的SM3哈希值与之前计算出的T’ = Hash(K || M || Pad || M’) 的值相一致，所不同的是，攻击者没有使用过密钥K=Passw0rd，说明攻击成功了。

**六、实验中的问题与总结：**

本次实验我基于Windows 10下安装GmSSL，以及基于Visual Studio使用C语言编程实现了对SM3哈希函数做长度扩展攻击的可展示实例。

实验中我遇到了几个问题。首先是安装GmSSL过程中遇到了一些错误，比如提示没有安装NASM、WIN32 Console模块、dmake以及没有设置好路径等，这些都在尝试中解决了。然后是攻击的环节，由于我参考了MD5的长度扩展攻击，而MD5填充的长度字段的存储方式是小端的，一开始对于SM3的填充我也用小端方式进行了，因此造成了攻击不成功，后来改为大端方式，解决了这一问题；另外，填充需要输入16进制数，一开始我没有用Notepad++的16进制插件输入，而是从网上复制的，造成编码可能有些问题，后来我用16进制插件输入，解决了这一问题。我还有一个在实际背景下应用的问题：实际情况中，一般是服务器生成密钥K，虽然攻击者不需要知道K的值，但也需要知道K的长度，这一长度的获取是否容易。

**【参考资料】**

[1] <https://baike.baidu.com/item/SM3/4421797?fr=aladdin> （SM3\_百度百科）.

[2] <https://blog.csdn.net/a344288106/article/details/80094878> （SM3密码杂凑算法实现及说明）.

[3] <https://blog.csdn.net/szuaurora/article/details/78125585> （MD5的长度扩展攻击详解，中文版）.

[4] <http://gmssl.org/docs/install.html> （GmSSL在Window下的编译和安装）.

[5] <https://github.com/iagox86/hash_extender>（MD5的长度扩展攻击详解，英文版）.