

**长度扩展攻击SM3**

学 院： 软件与微电子学院

专 业： 计算机技术

学生姓名： 吴宏凯

学 号： 1901210530

日 期： 2019-10-14

[**一、SM3算法介绍： 3**](#_Toc21950659)

[**二、长度扩展攻击原理： 4**](#_Toc21950660)

[**三、设计思路： 4**](#_Toc21950661)

[**四、实验结果： 5**](#_Toc21950662)

## 一、SM3算法介绍：

1. **概述：**

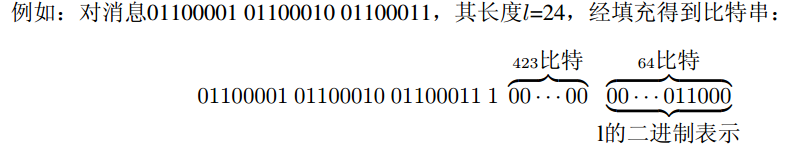
SM3是中华人民共和国政府采用的一种密码散列函数标准。在商用密码体系中，SM3主要用于数字签名及验证、消息认证码生成及验证、随机数生成等，其算法公开。据国家密码管理局表示，其安全性及效率与SHA-256相当。  
 此算法对输入长度小于2的64次方的比特消息，经过填充和迭代压缩，生成长度为256比特的杂凑值，其中使用了异或，模，模加，移位，与，或，非运算，由填充，迭代过程，消息扩展和压缩函数所构成。

**2、SM3算法与长度扩展攻击相关的步骤：**

1）补位、补长度

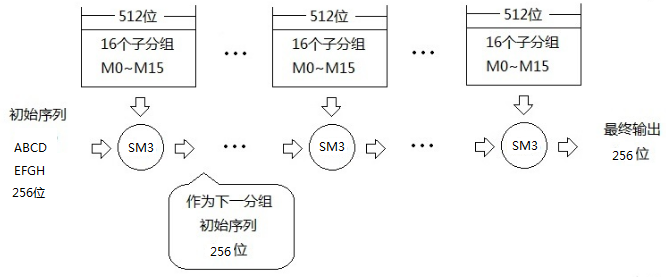
假设消息m 的长度为l 比特。首先将比特“ 1”添加到消息的末尾，再添加k 个“ 0”， k是满

足l + 1 + k ≡ 448mod512 的最小的非负整数。然后再添加一个64位比特串，该比特串是长度l的二进制表示。填充后的消息m′ 的比特长度为512的倍数



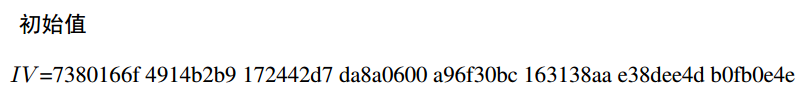
2）迭代压缩：

加密过程如下图



不需要关心每轮SM3的计算细节，只需要知道补位后的消息以512bit为单位分组开始进行加密计算，首个分组与初始链变量进行运算，得到256bit的值，该值作为新的链变量覆盖之前的链变量，以此类推，而最后一轮产生的就是我们计算出来的 sm3 值。

其中初始链变量为：



二、长度扩展攻击原理：

在不知道密文的情况下，推算出另一个匹配的哈希值。

1、适用对象：

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）。

满足上述要求的散列函数称为Merkle–Damgård散列函数，下列散列函数都属于Merkle–Damgård散列函数：

MD4

MD5

RIPEMD-160

SHA-0

SHA-1

SHA-256

SM3等

所以可以使用长度扩展攻击对SM3算法进行攻击。

2、预期达到的攻击效果：

对于H(salt+data)形式的加密，攻击者在不知道salt的前提下，可以通过控制data的值，获取H(salt+一定规则构造的data)，使两个hash值相等。

## 三、设计思路：

**1、攻击思路：**

由于SM3使用链变量覆盖的方式进行多轮加密，所以利用这一特性可以构成长度扩展攻击。通过常规加密后得到的hash值正是最后一轮加密后经过大端序排列后得到的链变量，在常规的加密之后，因为我们持有对数据修改的权限，所以按照一定的规则修改消息，做下一轮加密，此时只需要知道上一轮消息产生的链变量，就可以跳过了中间多个步骤，直接能在明文未知的情况下计算出附加随机值后的hash值，再与真实明文附加相同随机值求得的hash值做比较，相同则说明攻击成功。

**2、具体步骤**

1）计算原明文下的hash值，并获取明文的长度。例如我输入的是“abc”，长度为3，做补位和补长度标识，得到：

[x, x, x, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 24]

其中，xxx表示假设明文未知。

2）在初始链变量IV下开始做SM3算法，得到最终hash值：

66c7f0f462eeedd9d1f2d46bdc10e4e24167c4875cf2f7a2297da02b8f4ba8e0

消息长度 len1=3

3）长度扩展：在原始消息的补位后面增加附加值，我选择的是“test”，重新做补位和补长度，得到：

[x, x, x, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 24, 116, 101, 115, 116, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 32]

再次计算hash值，得到：

6cfb8ba0028953eee830c72c1aeeb1a68fd9edc7d70773dc974af49910821be4

4）构造虚假的消息并做补位：

根据原消息长度为3，选择用3个a来作为虚拟的明文，接着补位到448bit，最后补上长度标识。得到：

[97, 97, 97, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 24]

在长度标识后面加上步骤2）选择的附加值“test”，接着以从aaa到test的所有字段再做一次补位和补长度，得到：

str=[97, 97, 97, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 24, 116, 101, 115, 116, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 32]

5）构造链变量：

由2）得到的hash值通过大端序排列，得到：

A=0x6cfb8ba0 E=0x8fd9edc7

B=0x028953ee F=0xd70773dc

C=0xe830c72c G=0x974af499

D=0x1aeeb1a6 H=0x10821be4

代替初始链变量IV，赋给IV\_new，作为str计算hash值的初始链变量。

6）以IV\_new作为初始链变量，将str的最后一个分组与IV\_new的做运算，最后得到的hash值为

6cfb8ba0028953eee830c72c1aeeb1a68fd9edc7d70773dc974af49910821be4

若与3）中长度扩展完得到的hash值相同，则攻击成功。

## 四、实验结果：

原消息为：“abc”，附加消息为“test”，求得的hash值为y1

任取的消息为：“aaa”，附加消息为“test”，求得的hash值为y2

如下图所示，y1==y2，攻击成功。

