**（1）SHA算法原理**

安全散列算法（英语：Secure Hash Algorithm，缩写为SHA）是一个密码散列函数家族，是FIPS所认证的安全散列算法。能计算出一个数字消息所对应到的，长度固定的字符串（又称消息摘要）的算法。且若输入的消息不同，它们对应到不同字符串的机率很高。

SHA家族的五个算法，分别是SHA-1、SHA-224、SHA-256、SHA-384，和SHA-512，由美国国家安全局（NSA）所设计，并由美国国家标准与技术研究院（NIST）发布；是美国的政府标准。后四者有时并称为SHA-2。SHA-1在许多安全协定中广为使用，包括TLS和SSL、PGP、SSH、S/MIME和IPsec，曾被视为是MD5（更早之前被广为使用的杂凑函数）的后继者。但SHA-1的安全性如今被密码学家严重质疑；虽然至今尚未出现对SHA-2有效的攻击，它的算法跟SHA-1基本上仍然相似；因此有些人开始发展其他替代的杂凑算法。

下面选择SHA算法家族中的SHA256讲解原理。

1.常量初始化

SHA256算法中用到了8个哈希初值以及64个哈希常量。其中，SHA256算法的8个哈希初值如下：

h0=0x6a09e667

h1=0xbb67ae85

h2=0x3c6ef372

h=0xa54ff53a

h4=0x510e527f

h5=0x9b05688c

h6=0x1f83d9ab

h7=0x5be0cd19

这些初值是对自然数中前8个质数（2,3,5,7,11,13,17,19）的平方根的小数部分取前32bit而来。

在SHA256算法中，用到的64个常量如下：

428a2f98 71374491 b5c0fbcf e9b5dba5

3956c25b 59f111f1 923f82a4 ab1c5ed5

d807aa98 12835b01 243185be 550c7dc3

72be5d74 80deb1fe 9bdc06a7 c19bf174

e49b69c1 efbe4786 0fc19dc6 240ca1cc

2de92c6f 4a7484aa 5cb0a9dc 76f988da

983e5152 a831c66d b00327c8 bf597fc7

c6e00bf3 d5a79147 06ca6351 14292967

27b70a85 2e1b2138 4d2c6dfc 53380d13

650a7354 766a0abb 81c2c92e 92722c85

a2bfe8a1 a81a664b c24b8b70 c76c51a3

d192e819 d6990624 f40e3585 106aa070

19a4c116 1e376c08 2748774c 34b0bcb5

391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3

748f82ee 78a5636f 84c87814 8cc70208

90befffa a4506ceb bef9a3f7 c67178f2

和8个哈希初值类似，这些常量是对自然数中前64个质数(2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97…)的立方根的小数部分取前32bit而来。

2.信息预处理(pre-processing)

HA256算法中的预处理就是在想要Hash的消息后面补充需要的信息，使整个消息满足指定的结构。信息的预处理分为两个步骤：附加填充比特和附加长度

STEP1：附加填充比特

在报文末尾进行填充，使报文长度在对512取模以后的余数是448。填充是这样进行的：先补第一个比特为1，然后都补0，直到长度满足对512取模后余数是448。需要注意的是，信息必须进行填充，也就是说，即使长度已经满足对512取模后余数是448，补位也必须要进行，这时要填充512个比特。因此，填充是至少补一位，最多补512位。例：以信息“abc”为例显示补位的过程。a,b,c对应的ASCII码分别是97,98,99。于是原始信息的二进制编码为：01100001 01100010 01100011

补位第一步，首先补一个“1” ： 0110000101100010 01100011 1

补位第二步,补423个“0”：01100001 01100010 01100011 10000000 00000000 … 00000000

补位完成后的数据如下（为了简介用16进制表示）：

61626380 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000

为什么是448?

因为在第一步的预处理后，第二步会再附加上一个64bit的数据，用来表示原始报文的长度信息。而448+64=512，正好拼成了一个完整的结构。

STEP2：附加长度值

附加长度值就是将原始数据（第一步填充前的消息）的长度信息补到已经进行了填充操作的消息后面。wiki百科中给出的原文是：append length of message (before pre-processing), in bits, as 64-bit big-endian integer。SHA256用一个64位的数据来表示原始消息的长度。因此，通过SHA256计算的消息长度必须要小于$ 2^64 $，当然绝大多数情况这足够大了。长度信息的编码方式为64-bit big-endian integer。关于Big endian的含义，文末给出了补充。回到刚刚的例子，消息“abc”，3个字符，占用24个bit。因此，在进行了补长度的操作以后，整个消息就变成下面这样了（16进制格式）

61626380 00000000 00000000 00000000

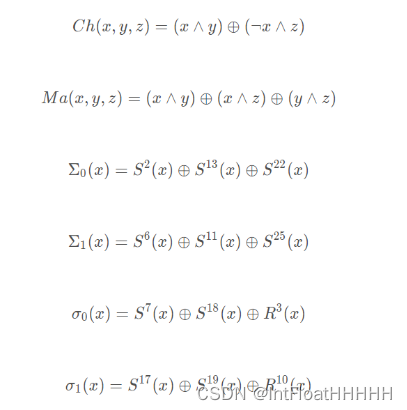
00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000018

3.逻辑运算

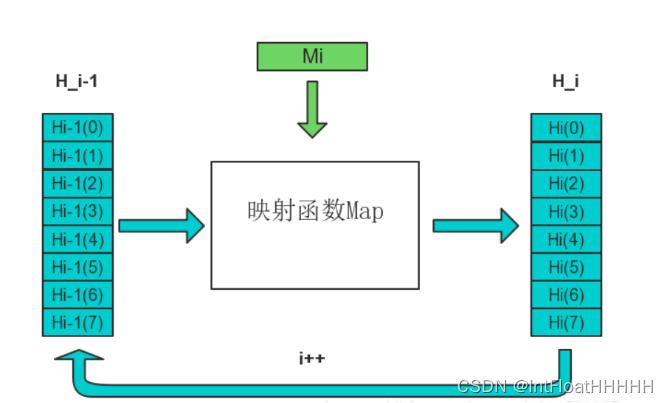
SHA256散列函数中涉及的操作全部是逻辑的位运算。包括如下的逻辑函数：





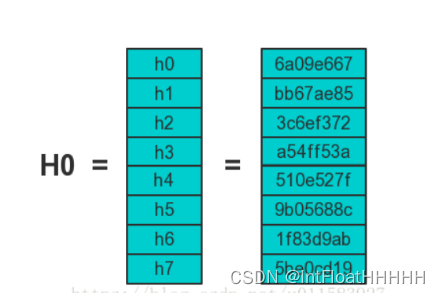
4.计算消息摘要

现在来介绍SHA256算法的主体部分，即消息摘要是如何计算的。首先：将消息分解成512-bit大小的块。假设消息M可以被分解为n个块，于是整个算法需要做的就是完成n次迭代，n次迭代的结果就是最终的哈希值，即256bit的数字摘要。一个256-bit的摘要的初始值H0，经过第一个数据块进行运算，得到H1，即完成了第一次迭代。H1经过第二个数据块得到H2，……，依次处理，最后得到Hn，Hn即为最终的256-bit消息摘要。将每次迭代进行的映射用$ Map(H\_{i-1}) = H\_{i} $表示，于是迭代可以更形象的展示为：



图中256-bit的Hi被描述8个小块，这是因为SHA256算法中的最小运算单元称为“字”（Word），一个字是32位。

此外，第一次迭代中，映射的初值设置为前面介绍的8个哈希初值，如下图所示：



下面开始介绍每一次迭代的内容，即映射$ Map(H\_{i-1}) = H\_{i} $的具体算法

STEP1：构造64个字（word）

break chunk into sixteen 32-bit big-endian words w[0], …, w[15]

对于每一块，将块分解为16个32-bit的big-endian的字，记为w[0], …, w[15]，也就是

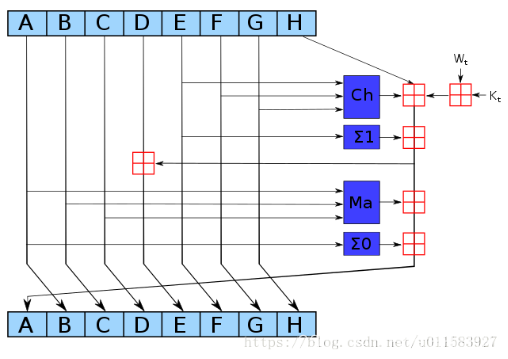
前16个字直接由消息的第i个块分解得到。其余的字由如下迭代公式得到：

IMG_256

STEP2：进行64次循环

映射 $ Map(H\_{i-1}) = H\_{i} $ 包含了64次加密循环

即进行64次加密循环即可完成一次迭代。每次加密循环可以由下图描述：



图中，ABCDEFGH这8个字（word）在按照一定的规则进行更新，其中深蓝色方块是事先定义好的非线性逻辑函数，上文已经做过铺垫。红色田字方块代表 mod $ 2^{32} $ addition，即将两个数字加在一起，如果结果大于$ 2^{32} ，你必须除以 ，你必须除以，你必须除以 2^{32} $并找到余数。ABCDEFGH一开始的初始值分别为$ H\_{i-1}(0),H\_{i-1}(1),…,H\_{i-1}(7) $。Kt是第t个密钥，对应我们上文提到的64个常量，Wt是本区块产生第t个word。原消息被切成固定长度512-bit的区块，对每一个区块，产生64个word，通过重复运行循环n次对ABCDEFGH这八个字循环加密。最后一次循环所产生的八个字合起来即是第i个块对应到的散列字符串$ H\_{i} $，由此完成了SHA256算法的所有介绍。