# 4.MD5散列算法

MD5算法全程是Message Digest Algorithm 5，中文名是消息摘要算法第五版，是一种散列算法，用来提供信息的完整性保护。

## 4.1.算法步骤

MD5算法以512位为一个分组处理输入的信息，每一个分组又被划分为16个32个子分组，经过一系列数学运算后，输出四个32位的分组数据，这4个32位的分组数据级联成128bit的散列。

### 4.1.1.填充

如果输入信息的长度(单位为bit！)对512求余的结果不等于448，就需要填充，使其长度对512求余结果为448。填充方法是一个bit的1和若干个bit的0.填充完成后，信息的长度满足：N\*512+448(bit)；

### 4.1.2.记录长度

用64bit的数据记录**填充之前**的数据长度，这64bit的数据紧随4.1.1中填充之后的数据，这样信息的长度就成为了：（N+1）\*512bit；

要注意的是，这里添加长度时的顺序一定是小端的，例如“abc”这个字符串，三个字节的长度(\0不计算在长度内)，换算成bit就是24，折合成十六进制就是0x18。按照正常的大端，这里要添加的就会是0x0000 0000 0000 0018，但是在MD5算法中要求的一定是：0x1800 0000 0000 0000的形式；

### 4.1.3.装入标准幻数

标准幻数(物理顺序)是：A=(0x01234567), B=(0x89ABCDEF), C=(0Xfedcba98), D=(0x76543210);

但是在程序中要注意，能够使用的应该是字节序相反的变量定义：A=(0x67452301), B=(0xefcdab89), C=(0x98badcfe), D=(0x10325476);

### 4.1.4.四轮循环运算

首先做如下一些基本函数定义：

F(X, Y, Z) = (X & Y) | ((~X) & Z);

G(X, Y, Z) = (X & Z) | (Y & (~Z));

H(X, Y, Z) = (X ^ Y ^ Z);

I(X, Y, Z) = Y ^ (X | (~Z));

这是四个非线性函数，这四个函数说明，如果X、Y和Z的对应位是独立和均匀的，那么结果的每一位也是独立和均匀的；

假设M[j]表示消息的第j个子分组，<<<表示循环左移，那么：

FF(a, b, c, d, M[j], s, ti)表示：a = b + ( (a + F(b, c, d) + M[j] + ti) <<< s );

GG(a, b, c, d, M[j], s, ti)表示：a = b + ( (a + G(b, c, d) + M[j] + ti) <<< s );

HH(a, b, c, d, M[j], s, ti)表示：a = b + ( (a + H(b, c, d) + M[j] + ti) <<< s );

II(a, b, c, d, M[j], s, ti)表示：a = b + ( (a + I(b, c, d) + M[j] + ti) <<< s );

1. for(i = 0; i < N+1; i++) //数据长度为(N+1)\*512bits，对每一个512bits的分组执行如下的操作
2. 将512bits分为16个分组，每个分组的size为32bits，定义这些分组为M[0]—M[15]，注意这些分组组合成32bit的数字的顺序也是小端的！例如，32bit的char是[0x12, 0x34, 0x56, 0x78];那么M[x]=0x78563412,而不是0x12345678！
3. 对这16个分组进行判断：

a = A; b = B; c = C; d = D;

//传说中的对M[j]的第一轮循环

        FF(a,b,c,d,M[0],7,0xd76aa478);

        FF(d,a,b,c,M[1],12,0xe8c7b756);

        FF(c,d,a,b,M[2],17,0x242070db);

        FF(b,c,d,a,M[3],22,0xc1bdceee);

        FF(a,b,c,d,M[4],7,0xf57c0faf);

        FF(d,a,b,c,M[5],12,0x4787c62a);

        FF(c,d,a,b,M[6],17,0xa8304613);

        FF(b,c,d,a,M[7],22,0xfd469501) ;

        FF(a,b,c,d,M[8],7,0x698098d8) ;

        FF(d,a,b,c,M[9],12,0x8b44f7af) ;

        FF(c,d,a,b,M[10],17,0xffff5bb1) ;

        FF(b,c,d,a,M[11],22,0x895cd7be) ;

        FF(a,b,c,d,M[12],7,0x6b901122) ;

        FF(d,a,b,c,M[13],12,0xfd987193) ;

        FF(c,d,a,b,M[14],17,0xa679438e) ;

        FF(b,c,d,a,M[15],22,0x49b40821);

        //传说中对M[j]的第二轮循环

        GG(a,b,c,d,M[1],5,0xf61e2562);

        GG(d,a,b,c,M[6],9,0xc040b340);

        GG(c,d,a,b,M[11],14,0x265e5a51);

        GG(b,c,d,a,M[0],20,0xe9b6c7aa) ;

        GG(a,b,c,d,M[5],5,0xd62f105d) ;

        GG(d,a,b,c,M[10],9,0x02441453) ;

        GG(c,d,a,b,M[15],14,0xd8a1e681);

        GG(b,c,d,a,M[4],20,0xe7d3fbc8) ;

        GG(a,b,c,d,M[9],5,0x21e1cde6) ;

        GG(d,a,b,c,M[14],9,0xc33707d6) ;

        GG(c,d,a,b,M[3],14,0xf4d50d87) ;

        GG(b,c,d,a,M[8],20,0x455a14ed);

        GG(a,b,c,d,M[13],5,0xa9e3e905);

        GG(d,a,b,c,M[2],9,0xfcefa3f8) ;

        GG(c,d,a,b,M[7],14,0x676f02d9) ;

        GG(b,c,d,a,M[12],20,0x8d2a4c8a);

        //传说中对M[j]的第三轮循环

        HH(a,b,c,d,M[5],4,0xfffa3942);

        HH(d,a,b,c,M[8],11,0x8771f681);

        HH(c,d,a,b,M[11],16,0x6d9d6122);

        HH(b,c,d,a,M[14],23,0xfde5380c) ;

        HH(a,b,c,d,M[1],4,0xa4beea44) ;

        HH(d,a,b,c,M[4],11,0x4bdecfa9) ;

        HH(c,d,a,b,M[7],16,0xf6bb4b60) ;

        HH(b,c,d,a,M[10],23,0xbebfbc70);

        HH(a,b,c,d,M[13],4,0x289b7ec6);

        HH(d,a,b,c,M[0],11,0xeaa127fa);

        HH(c,d,a,b,M[3],16,0xd4ef3085);

        HH(b,c,d,a,M[6],23,0x04881d05);

        HH(a,b,c,d,M[9],4,0xd9d4d039);

        HH(d,a,b,c,M[12],11,0xe6db99e5);

        HH(c,d,a,b,M[15],16,0x1fa27cf8) ;

        HH(b,c,d,a,M[2],23,0xc4ac5665);

        //传说中对M[j]的第四轮循环

        II(a,b,c,d,M[0],6,0xf4292244) ;

        II(d,a,b,c,M[7],10,0x432aff97) ;

        II(c,d,a,b,M[14],15,0xab9423a7);

        II(b,c,d,a,M[5],21,0xfc93a039) ;

        II(a,b,c,d,M[12],6,0x655b59c3) ;

        II(d,a,b,c,M[3],10,0x8f0ccc92) ;

        II(c,d,a,b,M[10],15,0xffeff47d);

        II(b,c,d,a,M[1],21,0x85845dd1) ;

        II(a,b,c,d,M[8],6,0x6fa87e4f) ;

        II(d,a,b,c,M[15],10,0xfe2ce6e0);

        II(c,d,a,b,M[6],15,0xa3014314) ;

        II(b,c,d,a,M[13],21,0x4e0811a1);

        II(a,b,c,d,M[4],6,0xf7537e82) ;

        II(d,a,b,c,M[11],10,0xbd3af235);

        II(c,d,a,b,M[2],15,0x2ad7d2bb);

        II(b,c,d,a,M[9],21,0xeb86d391);

        A += a;

        B += b;

        C += c;

        D += d;

1. 对所有子分组执行如上的循环操作；

### 4.1.5.最终结果

处理完所有512分组的内容后，得到一组新的A B C D的值，将这些值按照ABCD的级联顺序排序，然后输出即为最终的MD5值。

需要注意的是，MD5是小端序列，所以输出时要做一次转换，例如A有32bit，分为A1A2A3A4，那么输出的就应该是A4A3A2A1。

### 4.1.6.一些典型的MD5值

md5(“”) = d41d8cd98f00b204e9800998ecf8427e

MD5 ("a") = 0cc175b9c0f1b6a831c399e269772661

MD5 (“ab”) = 187ef4436122d1cc2f40dc2b92f0eba0

MD5 ("abc") = 900150983cd24fb0d6963f7d28e17f72

MD5 (“abcde”) = ab56b4d92b40713acc5af89985d4b786

MD5 ("message digest") = f96b697d7cb7938d525a2f31aaf161d0

MD5 ("abcdefghijklmnopqrstuvwxyz") = c3fcd3d76192e4007dfb496cca67e13b

MD5 ("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz") = f29939a25efabaef3b87e2cbfe641315