南開大學
Nankai University

南　　开　　大　　学

网 络 空 间 安 全 学 院

密码学实验

Hash 函数 MD5

学号：2013018

姓名：许健

年级：2020 级

专业：信息安全

2022 年 12 月 28 日

目录

一、 实验目的

通过实际编程了解 MD5 算法的过程，加深对 Hash 函数的认识。

二、 MD5 算法

(一) hash 函数原理

Hash 函数是将任意长的数字串转换成一个较短的定长输出数字串的函数，输出的结果称为 Hash 值。Hash 函数具有如下特点：

1. 快速性：对于任意一个输入值 x，由 Hash 函数，计算 Hash 值 y，即是非常容易的。
2. 单向性：对于任意一个输出值 y，希望反向推出输入值 x，使得，是非常困难的。
3. 无碰撞性：包括强无碰撞性和弱无碰撞性，一个好的 Hash 函数应该满足强无碰撞性，即找到两个不同的数字串 x 和 y，满足，在计算上是不可能的。

Hash 函数可用于数字签名、消息的完整性检验。消息的来源认证检测等。现在常用的 Hash 算法有 MD5、SHA － 1 等。下面从 MD5 入手来介绍 Hash 算法的实现机制。

(二) MD5 算法实现

MD 系列单向散列函数是由 Ron Rivest 设计的，MD5 算法对任意长度的输入值处理后产生 128 位的 Hash 值。MD5 算法的实现步骤如图1所示。

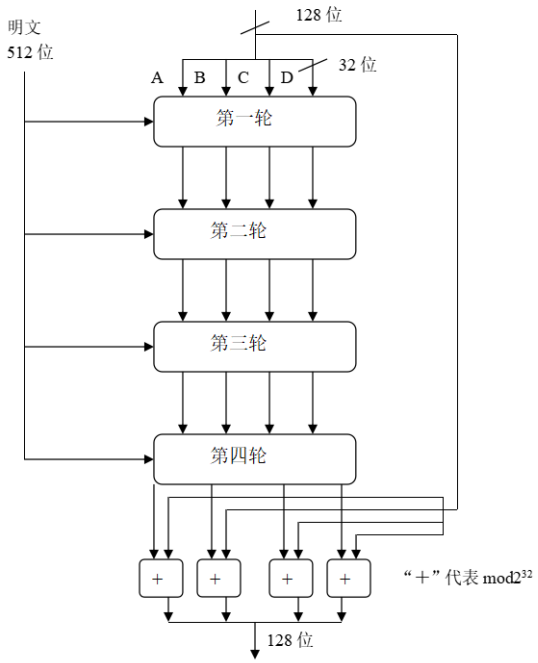

图 1: MD5 算法流程图

在 MD5 算法中，首先需要对信息进行填充，使其字节长度与 448 模 512 同余，即信息的字节长度扩展至 n*512+448，n 为一个正整数。填充的方法如下：在信息的后面填充第一位为 1，其余各位均为 0，直到满足上面的条件时才停止用 0 对信息的填充。然后，再在这个结果后面附加一个以 64 位二进制表示的填充前信息长度。经过这两步的处理，现在的信息字节长度为 n*512+448+64=(n+1)*512，即长度恰好是 512 的整数倍，这样做的目的是为了满足后面处理中对信息长度的要求。

MD5 中有 A、B、C、D，4 个 32 位被称为链接变量的整数参数，它们的初始值分别为：A0 = 0x01234567，B0 = 0x89abcdef，C0=0xfedcba98，D0=0x76543210。当设置好这 4 个链接变量后，就开始进入算法的 4 轮循环运算。循环的次数是信息中 512 位信息分组数目。

首先将上面 4 个链接变量复制到变量 A、B、C、D 中，以备后面进行处理。然后进入主循环，主循环有 4 轮，每轮循环都很相似。第一轮进行 16 次操作，每次操作对 A、B、C、D 中的 3 个做一次非线性函数运算，然后将所得结果加上第四个变量，文本的一个子分组（32 位）和一个常数。再将所得结果向左循环移 S 位，并加上 A、B、C、D 其中之一。最后用该结果取代 A、B、C、D 其中之一。

以下是每次操作中用到的 4 个非线性函数（每轮一个）。

$$
\begin{aligned}
F(B,C,D) &= (B \wedge C) \vee (\bar{B} \wedge D) \\
G(B,C,D) &= (B \wedge D) \vee (C \wedge \bar{D}) \\
H(B,C,D) &= B \oplus C \oplus D \\
I(B,C,D) &= C \oplus (B \vee \bar{D})
\end{aligned}
$$

MD5 轮主要操作为：

$$a = b + ((a + f(b,c,d) + M + t) << s)$$

对应于四轮操作，f 分别取 F，G，H，I；对每一轮的 16 次运算，M 分别取 M1，M2，…，M16。对于 4 轮共 64 次运算，t 为给定的一些常数。所有这些操作完成之后，将 A，B，C，D 分别加上 A0，B0，C0，D0。然后用下一分组数据继续进行运算，最后得到一组 A，B，C，D。把这组数据级联起来，即得到 128 比特的 Hash 结果。

三、 实验内容

(一) 程序流程图

MD5 编码函数首先会将字符串填充,存放在向量 vector 中。每轮会调用迭代函数 iterateFunc，处理 512bit，处理完后返回格式化输出的迭代变量 A、B、C、D。

迭代函数 iterateFunc 内部又会分为 64 轮，根据轮次选择函数 F、G、H、I，不断更新迭代变量 A、B、C、D 的值。

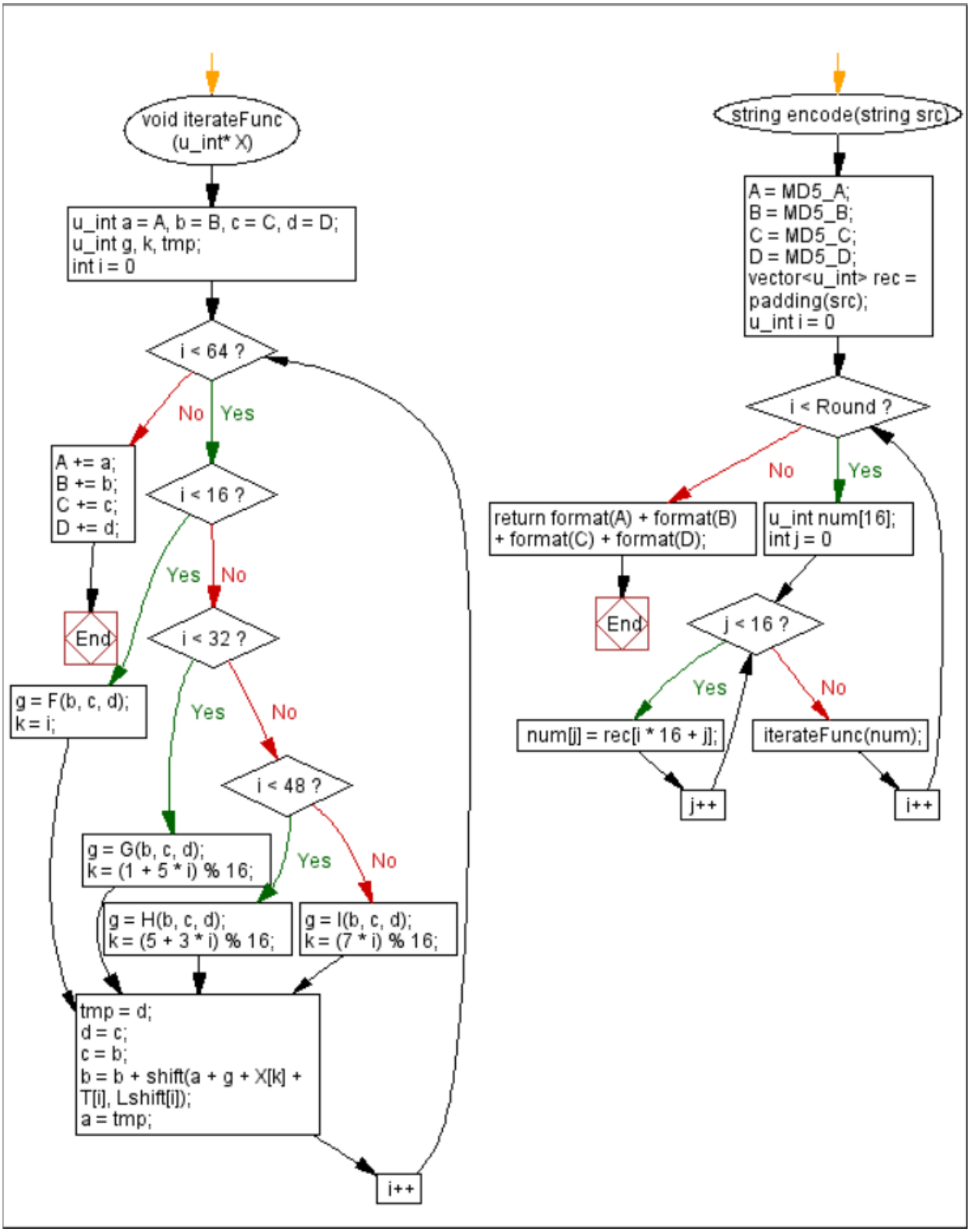

图 2: 程序流程图

(二) MD5 编码函数

MD5 编码函数, 首先初始化迭代变量 A、B、C、D, 然后填充输入字符串并保存到向量 vector 中, 之后进行 Round 轮迭代, 每次迭代使用 512 字节, 调用 iterateFunc 函数。最后将迭代结果保存到 A、B、C、D 中, 调用 format 函数格式化输出。

encode 函数

```
string encode(string src) {
    A = MD5_A;
    B = MD5_B;
    C = MD5_C;
    D = MD5_D;
    vector<u_int> rec = padding(src);
    for (u_int i = 0; i < Round; i++) {
        u_int num[16];
        for (int j = 0; j < 16; j++) {
            num[j] = rec[i * 16 + j];
        }
        iterateFunc(num);
    }
    return format(A) + format(B) + format(C) + format(D);
}
```

(三) 迭代函数

MD5 的迭代函数，根据传进去的 512 字节，更新 A、B、C、D，一共会被调用 Round 次。调用了循环移位函数 shift，并且函数 f 会根据循环次数选择 F、G、H、I 中其中一个。

迭代函数

```
void iterateFunc(u_int* X) {
    u_int a = A, b = B, c = C, d = D;
    u_int g, k, tmp;
    for (int i = 0; i < 64; i++) {
        if (i < 16) {
            g = F(b, c, d);
            k = i;
        }
        else if (i < 32) {
            g = G(b, c, d);
            k = (1 + 5 * i) % 16;
        }
        else if (i < 48) {
            g = H(b, c, d);
            k = (5 + 3 * i) % 16;
        }
        else {
            g = I(b, c, d);
            k = (7 * i) % 16;
        }
        tmp = d;
        d = c;
        c = b;
        b = b + shift(a + g + X[k] + T[i], Lshift[i]);
```

```
            a = tmp;
        }
        A += a;
        B += b;
        C += c;
        D += d;
}
```

（四） 格式化输入输出

padding 函数负责将输入的字符串进行填充，满足 MD5 格式要求，并将 bit 存储到 vector 中返回，vector 向量每个元素存储 32 位，方便后续的计算。

format 函数负责将 32bit 的数据转换成 string 字符串返回输出，用来将迭代多轮的 A、B、C、D 合成一个 string 输出。

输入填充 padding

```
vector<u_int> padding(string src) {
    Round = ((src.length() + 8) / 64) + 1;
    vector<u_int> rec(Round * 16);
    for (u_int i = 0; i < src.length(); i++) {
        rec[i >> 2] |= (int)(src[i]) << ((i % 4) * 8);
    }
    rec[src.length() >> 2] |= (0x80 << ((src.length() % 4) * 8));
    rec[rec.size() - 2] = (src.length() << 3);
    return rec;
}
```

格式化输出 format

```
string format(u_int num) {
    string res = "";
    const char str[] = "0123456789abcdef";
    u_int base = 1 << 8;
    for (int i = 0; i < 4; i++) {
        string tmp = "";
        u_int b = (num >> (i * 8)) % base & 0xff;
        for (int j = 0; j < 2; j++) {
            tmp = str[b % 16] + tmp;
            b /= 16;
        }
        res += tmp;
    }
    return res;
}
```

（五） 辅助函数

主要包括循环移位 shift 函数，以及 F、G、H、I 函数，用于轮迭代计算。

辅助函数

```
u_int F(u_int b, u_int c, u_int d) {
    return (b & c) | ((~b) & d);
}
u_int G(u_int b, u_int c, u_int d) {
    return (b & d) | (c & (~d));
}
u_int H(u_int b, u_int c, u_int d) {
    return b ^ c ^ d;
}
u_int I(u_int b, u_int c, u_int d) {
    return c ^ (b | (~d));
}
u_int shift(u_int a, u_int n) {
    return (a << n) | (a >> (32 - n));
}
```

四、 实验展示

从 MD5 哈希函数的执行结果来看，无论输入是空字符串还是多长，最终的输出都是 128 比特。并且输出的结果完全正确。

```
Text:
MD5: d41d8cd98f00b204e9800998ecf8427e
Text: a
MD5: 0cc175b9c0f1b6a831c399e269772661
Text: abc
MD5: 900150983cd24fb0d6963f7d28e17f72
Text: message digest
MD5: f96b697d7cb7938d525a2f31aaf161d0
Text: abcdefghijklmnopqrstuvwxyz
MD5: c3fcd3d76192e4007dfb496cca67e13b
Text: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
MD5: d174ab98d277d9f5a5611c2c9f419d9f
Text: 12345678901234567890123456789012345678901234567890123456789012345678901234567890
MD5: 57edf4a22be3c955ac49da2e2107b67a
Text:
```

图 3: MD5 结果展示

五、 雪崩效应检测

接下来我们对 MD5 哈希函数进行雪崩效应检测，从之前的结果展示中可以看出，对于输出为 12345678901234567890123456789012345678901234567890123456789012345678901234567890 的字符串，最终会被加密成 57edf4a22be3c955ac49da2e2107b67a。我们尝试在每个 0 后面填充一个空格，一共进行 8 次测试，看看新的 hash 字符串和之前的 hash 字符串对比会有多少位不同。

```
Text: 12345678901234567890123456789012345678901234567890123456789012345678901234567890
MD5: 57edf4a22be3c955ac49da2e2107b67a
Text: 1234567890 12345678901234567890123456789012345678901234567890123456789012345678901234567890
MD5: d1caecfc6acb077981fea866efdf4b2b
改变位数：61
Text: 12345678901234567890 12345678901234567890123456789012345678901234567890123456789012345678901234567890
MD5: 3594acc7186920b36ec2b3b51414c15d
改变位数：65
Text: 123456789012345678901234567890 12345678901234567890123456789012345678901234567890123456789012345678901234567890
MD5: 16e6711e6180c8c3698c83752b7757a9
改变位数：56
Text: 123456789012345678901234567890123456789012345678901234567890 12345678901234567890123456789012345678901234567890123456789012345678901234567890
MD5: b2eb6f504653aab4f5cef4b7adaee6c1
改变位数：64
Text: 123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890 12345678901234567890123456789012345678901234567890123456789012345678901234567890
MD5: fa7a19d7254de1df3cf614dca764ef7c
改变位数：66
Text: 123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890 12345678901234567890123456789012345678901234567890123456789012345678901234567890
MD5: 5ea7c31a25c3fe694628ad12fa0855a1
改变位数：66
Text: 123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890 1234567890
MD5: 1fd5c74c44321e25a411c26388fa1762
改变位数：60
Text: 12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
MD5: d5a01d2d92d9026419f2c4bb5a35b08a
改变位数：67
Text:
```

图 4: 雪崩效应检测

从检测的结果来看，8 次 hash 改变的位数分别为 61、65、56、64、66、66、60、67，大概平均每次有一半的位数（64 位）会发生变化，我们说 MD5 哈希算法可以有效防止从输出推测输入，从而导致该算法部分乃至整个算法被全部破解。