南開大學
Nankai University

南　　开　　大　　学

网 络 空 间 安 全 学 院

密码学实验报告

Hash 函数 MD5

于文明

年级：2020 级

专业：信息安全

指导教师：古力

2022 年 12 月 29 日

摘要

关键字：Parallel

目录

一、 实验内容

(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) 实验环境

运行 Windows 操作系统的 PC 机，具有 VC 等语言编译环境

(三) 实验内容

1、算法分析

请参照教材内容，分析 MD5 算法实现的每一步原理。

2、算法实现：

利用 Visual C ++语言，自己编写 MD5 的实现代码，并检验代码实现的正确性。

3、雪崩效应检验：

尝试对一个长字符串进行 Hash 运算，并获得其运算结果。对该字符串进行轻微的改动，比如增加一个空格或标点，比较 Hash 结果值的改变位数。进行 8 次这样的测试。

(四) 实验要求

1、自己编写完整的 MD5 实现代码，并提交程序和程序流程图。

2. 对编好的 MD5 算法，测试其雪崩效应，要求给出文本改变前和改变后的 Hash 值，并计算出 6 改变的位数。写出 8 次测试的结果，并计算出平均改变的位数

NKU

二、 程序流程图

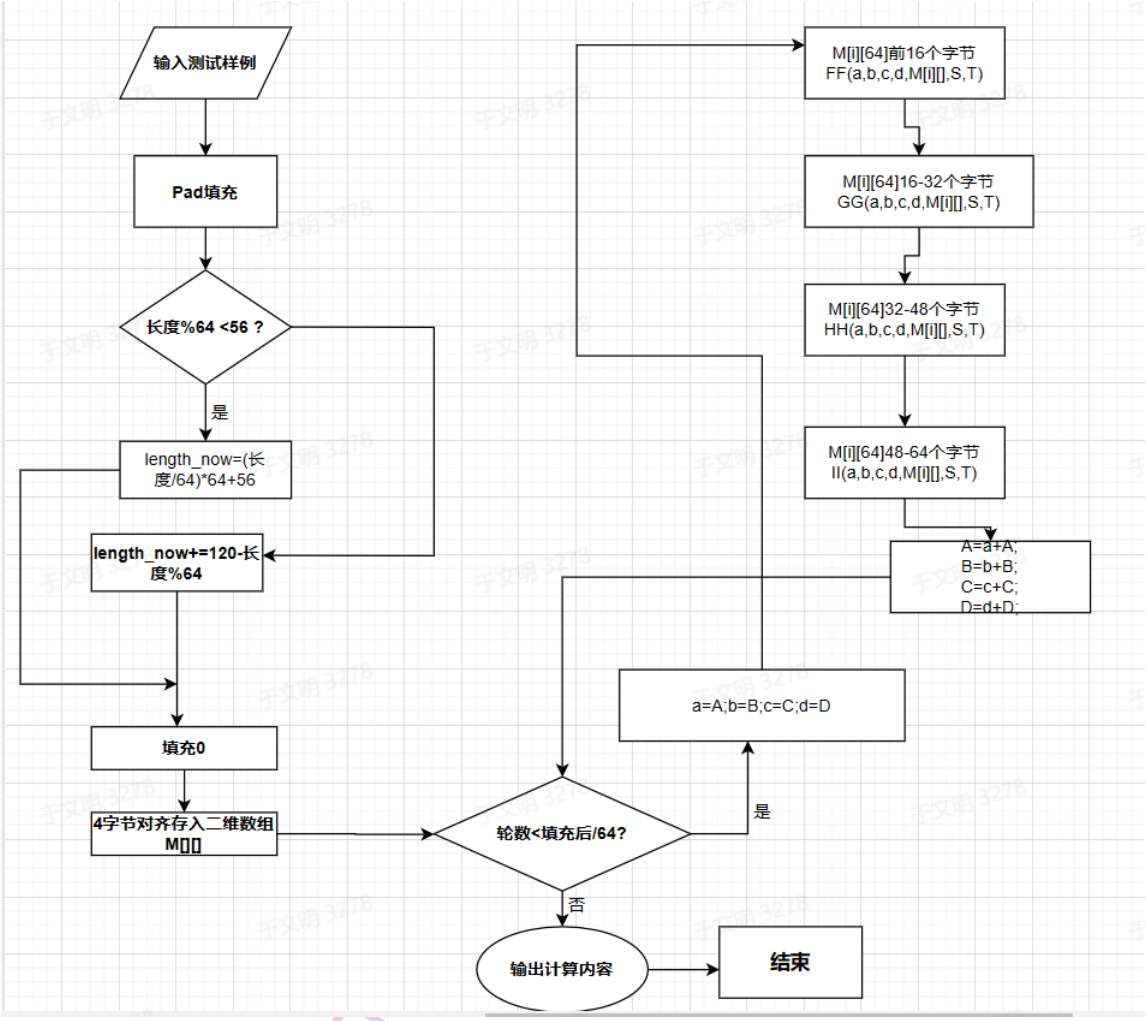

图 1: md5 程序流程图

三、 MD5 算法介绍

(一) 填充

在 MD5 算法中，首先需要对信息进行填充，使其位长对 512 求余的结果等于 448，并且填充必须进行，即使其位长对 512 求余的结果等于 448。因此，信息的位长（Bits Length）将被扩展至 N*512+448，N 为一个非负整数，N 可以是零。

填充的方法如下:

1) 在信息的后面填充一个 1 和无数个 0，直到满足上面的条件时才停止用 0 对信息的填充。

2) 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（单位为 Bit），如果二进制表示的填充前信息长度超过 64 位，则取低 64 位。

经过这两步的处理，信息的位长 =N*512+448+64=(N+1）*512，即长度恰好是 512 的整数倍。这样做的原因是为满足后面处理中对信息长度的要求。

(二) 初始化变量

初始的 128 位值为初试链接变量，这些参数用于第一轮的运算，以大端字节序来表示，他们分别为：A=0x01234567，B=0x89ABCDEF，C=0xFEDCBA98，D=0x76543210。

（每一个变量给出的数值是高字节存于内存低地址，低字节存于内存高地址，即大端字节序。在程序中变量 A、B、C、D 的值分别为 0x67452301, 0xEFCDAB89, 0x98BADCFE, 0x10325476）

(三) 处理分组数据

每一分组的算法流程如下:

第一分组需要将上面四个链接变量复制到另外四个变量中：A 到 a，B 到 b，C 到 c，D 到 d。从第二分组开始的变量为上一分组的运算结果，即 A = a，B = b，C = c，D = d。

主循环有四轮（MD4 只有三轮），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 和 d 中的其中三个作一次非线性函数运算，然后将所得结果加上第四个变量，文本的一个子分组和一个常数。再将所得结果向左环移一个不定的数，并加上 a、b、c 或 d 中之一。最后用该结果取代 a、b、c 或 d 中之一。

以下是每次操作中用到的四个非线性函数（每轮一个）。

F(X ,Y ,Z) = (X Y) | ((X) Z)

G(X ,Y ,Z) = (X Z) | (Y (Z))

H(X ,Y ,Z) =X ^{YZ}

I(X ,Y ,Z) =Y〔X|(Z))

(是与（And)，| 是或（Or)， 是非（Not)， Xor

这四个函数的说明：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应是独立和均匀的。

F 是一个逐位运算的函数。即，如果 X，那么 Y，否则 Z。函数 H 是逐位奇偶操作符。

假设 Mj 表示消息的第 j 个子分组（从 0 到 15)，常数 ti 是 4294967296*abs(sin(i)) 的整数部分，i 取值从 1 到 64，单位是弧度。(4294967296=2 的 32 次方)

现定义:

FF(a ,b ,c ,d ,Mj ,s ,ti) 操作为 a = b + ((a + F(b,c,d) + Mj + ti) « s)

GG(a ,b ,c ,d ,Mj ,s ,ti) 操作为 a = b + ((a + G(b,c,d) + Mj + ti) « s)

HH(a ,b ,c ,d ,Mj ,s ,ti) 操作为 a = b + ((a + H(b,c,d) + Mj + ti) « s)

II(a ,b ,c ,d ,Mj ,s ,ti) 操作为 a = b + ((a + I(b,c,d) + Mj + ti) « s)

现定义:

FF(a ,b ,c ,d ,Mj ,s ,ti) 操作为 a = b + ((a + F(b,c,d) + Mj + ti) « s)

GG(a ,b ,c ,d ,Mj ,s ,ti) 操作为 a = b + ((a + G(b,c,d) + Mj + ti) « s)

HH(a ,b ,c ,d ,Mj ,s ,ti) 操作为 a = b + ((a + H(b,c,d) + Mj + ti) « s)

II(a ,b ,c ,d ,Mj ,s ,ti) 操作为 a = b + ((a + I(b,c,d) + Mj + ti) « s)

注意：“«”表示循环左移位，不是左移位。

(四) 输出

最后的输出是 a、b、c 和 d 的级联。

NKU

四、 核心代码

(一) 辅助函数 (十六进制转二进制)

伪代码

itob

```
string htob(string b)
{
        string result;
        for (int i = 0; i < 32; i++)
        {
                //cout << i << endl;
                if (b[i] == '0')
                {
                        result += "0000";
                }
                if (b[i] == '1')
                {
                        result += "0001";
                }
                if (b[i] == '2')
                {
                        result += "0010";
                }
                if (b[i] == '3')
                {
                        result += "0011";
                }
            ...
        }

        return result;
}
```

NKU

(二) MD5 类

1. 类框架 (md5.h)

itob

```
class MD5
{
        //unsigned char是8bit, 1字节; unsigned int是32bit, 1个字
        unsigned int input[16];  //每个分组512比特,16个字
        unsigned int buffer[4];          //4个32比特字ABCD

public:
        MD5();
```

```
        string  pad(const char* message); //填充
        string round(unsigned int input[16]);    //4轮处理
        char* transfer(unsigned int B);
};
```

2. 填充函数 pad

itob

```
string  MD5::pad(const char* message)
{

        int len = strlen(message);
        //x % 512 = 64;
        int lnow;
        if (len * 8 % 448 == 0)
        {
                lnow = len * 8 + 512 + 64;
        }
        else
        {
                lnow = (((len * 8 + 64) / 512) + 1) * 512 - 64 + 64;
        }
        int totalgroup = lnow / 512;
        //cout << totalgroup << "个消息分组" << endl;
        unsigned int* group = new unsigned int[totalgroup * 16];
        memset(group, 0, sizeof(unsigned int) * totalgroup * 16);

        for (int i = 0; i < len; i++)
        {
                group[i >> 2] |= message[i] << ((i % 4) * 8);
                //cout << group[i] ;
        }

        group[len >> 2] |= 0x80 << ((len % 4) * 8);
        group[totalgroup * 16 - 2] = len * 8;


        /*for (int i = 0; i < totalgroup * 16; i++)
        {
        cout << group[i];
        }
        cout << endl;*/

        string temp[100];
        for (int i = 0; i < totalgroup; i++)
        {
                unsigned int input[16];
```

NKU

```
40                  for (int j = 0; j < 16; j++)
41                  {
42                          input[j] = group[j + i * 16];
43                  }
44                  temp[i] = round(input);
45          }
46          return temp[totalgroup - 1];
47
48  }
```

3. 类函数 round

itob

```
1   string MD5::round(unsigned int input[16])
2   {
3           unsigned int a = buffer[0];
4           unsigned int b = buffer[1];
5           unsigned int c = buffer[2];
6           unsigned int d = buffer[3];
7
8           for (int i = 0; i < 64; i++)
9           {
10                  if (i >= 0 && i < 16)
11                  {
12                          unsigned f = F(b, c, d);
13
14                          unsigned temp = d;
15                          d = c;
16                          c = b;
17                          b = b + moveleft(f + a + input[i] + T[i], s[i]);
18                          a = temp;
19
20
21                  }
22                  if (i >= 16 && i < 32)
23                  {
24                          unsigned g = G(b, c, d);
25                          unsigned temp = d;
26                          d = c;
27                          c = b;
28                          b = b + moveleft(g + a + input[(1 + 5 * i) % 16] + T[
                                i], s[i]);
29                          a = temp;
30                  }
31                  if (i >= 32 && i < 48)
32                  {
33                          unsigned h = H(b, c, d);
```

NKU

```
34                      unsigned temp = d;
35                      d = c;
36                      c = b;
37                      b = b + moveleft(h + a + input[(5 + 3 * i) % 16] + T[
                            i], s[i]);
38                      a = temp;
39              }
40              if (i >= 48 && i < 64)
41              {
42                      unsigned ii = I(b, c, d);
43                      unsigned temp = d;
44                      d = c;
45                      c = b;
46                      b = b + moveleft(ii + a + input[(7 * i) % 16] + T[i],
                             s[i]);
47                      a = temp;
48              }
49
50      }
51      buffer[0] += a;
52      buffer[1] += b;
53      buffer[2] += c;
54      buffer[3] += d;
55
56
57      //cout << buffer[0] + buffer[1] + buffer[2] + buffer[3]<<endl;
58      char bufa[9], bufb[9], bufc[9], bufd[9];
59      char buffa[9], buffb[9], buffc[9], buffd[9];
60      char bufA[9], bufB[9], bufC[9], bufD[9];
61      itoa(buffer[0], bufa, 16);   //大端小端的问题
62      itoa(buffer[1], bufb, 16);
63      itoa(buffer[2], bufc, 16);
64      itoa(buffer[3], bufd, 16);
65
66      if (strlen(bufa) != 8)
67      {
68              bufa[7] = '0';
69      }
70      else
71      {
72              for (int i = 0; i < 9; i++)
73                      buffa[i] = bufa[i];
74      }
75
76
77      if (strlen(bufb) != 8)
78      {
79              int x = 8 - strlen(bufb);
```

NKU

```
80                  for (int i = 0; i < x; i++)
81                          buffb[i] = '0';
82                  for (int i = 0; i < 9 - x; i++)
83                          buffb[i + x] = bufb[i];
84          }
85          else
86          {
87                  for (int i = 0; i < 9; i++)
88                          buffb[i] = bufb[i];
89          }
90
91
92
93          if (strlen(bufc) != 8)
94          {
95                  bufc[7] = '0';
96          }
97          else
98          {
99                  for (int i = 0; i < 9; i++)
100                         buffc[i] = bufc[i];
101         }
102
103
104         if (strlen(bufd) != 8)
105         {
106                 bufd[7] = '0';
107         }
108         else
109         {
110                 for (int i = 0; i < 9; i++)
111                         buffd[i] = bufd[i];
112         }
113
114         /*cout << buffa << endl;
115         cout << buffb << endl;
116         cout << buffc << endl;
117         cout << buffd << endl;*/
118         //a
119         bufA[0] = buffa[6];
120         bufA[1] = buffa[7];
121         bufA[2] = buffa[4];
122         bufA[3] = buffa[5];
123         bufA[4] = buffa[2];
124         bufA[5] = buffa[3];
125         bufA[6] = buffa[0];
126         bufA[7] = buffa[1];
127         //b
```

NKU

```
128         bufB[0] = buffb[6];
129         bufB[1] = buffb[7];
130         bufB[2] = buffb[4];
131         bufB[3] = buffb[5];
132         bufB[4] = buffb[2];
133         bufB[5] = buffb[3];
134         bufB[6] = buffb[0];
135         bufB[7] = buffb[1];
136         //c
137         bufC[0] = buffc[6];
138         bufC[1] = buffc[7];
139         bufC[2] = buffc[4];
140         bufC[3] = buffc[5];
141         bufC[4] = buffc[2];
142         bufC[5] = buffc[3];
143         bufC[6] = buffc[0];
144         bufC[7] = buffc[1];
145         //d
146         bufD[0] = buffd[6];
147         bufD[1] = buffd[7];
148         bufD[2] = buffd[4];
149         bufD[3] = buffd[5];
150         bufD[4] = buffd[2];
151         bufD[5] = buffd[3];
152         bufD[6] = buffd[0];
153         bufD[7] = buffd[1];
154
155         string result;
156         for (int i = 0; i < 8; i++)
157         {
158
159                 //cout << bufA[i];
160                 result += bufA[i];
161         }
162         for (int i = 0; i < 8; i++)
163         {
164                 //cout << bufB[i];
165                 result += bufB[i];
166         }
167         for (int i = 0; i < 8; i++)
168         {
169                 //cout << bufC[i];
170                 result += bufC[i];
171         }
172         for (int i = 0; i < 8; i++)
173         {
174                 //cout << bufD[i];
175                 result += bufD[i];
```

NKU

```
        }

        //cout << endl;
        return result;

}
```

4. 大小端转换 transfer

itob

```
char* MD5::transfer(unsigned int B)
{
    char bufa[9];
    char buffa[9];
    char bufA[9];

    itoa(B, bufa, 16);    //转化为16进制

    if (strlen(bufa) != 8)  //补零
    {
        int x = 8 - strlen(bufa);
        for (int i = 0; i < x; i++)
            buffa[i] = '0';
        for (int i = 0; i < 9 - x; i++)
            buffa[i + x] = bufa[i];
    }
    else
    {
        for (int i = 0; i < 9; i++)
            buffa[i] = bufa[i];
    }

    bufA[0] = buffa[6];    //大端小端问题
    bufA[1] = buffa[7];
    bufA[2] = buffa[4];
    bufA[3] = buffa[5];
    bufA[4] = buffa[2];
    bufA[5] = buffa[3];
    bufA[6] = buffa[0];
    bufA[7] = buffa[1];

    return bufA;
}
```

(三) 主函数实现

NKU

itob

```
int main(int argc, char* argv[]) {


    char test;
    printf("是否继续测试, 如果是, 请输入C ,否则请输入Q :\n");
    scanf("%c", &test);
    if (test == 'C')
    {
        for (int i = 0; i < 15; i++)
        {
            if (i == 7)
            {
                cout << "-----------------SNOW TEST--------------------" <<
                    endl;
            }
            cout << "TEST" << i + 1 << " " << endl;
            char* message = tests[i].msg;
            cout << "MESSAGE IS:  " << message << endl;
            char buffer[100];
            memset(buffer, 0, sizeof(buffer));
            sprintf(buffer, "%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X
                %02X%02X%02X%02X",
                tests[i].hash[0], tests[i].hash[1], tests[i].hash[2], tests[i
                    ].hash[3],
                tests[i].hash[4], tests[i].hash[5], tests[i].hash[6], tests[i
                    ].hash[7],
                tests[i].hash[8], tests[i].hash[9], tests[i].hash[10], tests[
                    i].hash[11],
                tests[i].hash[12], tests[i].hash[13], tests[i].hash[14],
                    tests[i].hash[15]);
            for (int i = 0; i < strlen(buffer); i++)
            {
                buffer[i] = tolower(buffer[i]);
            }
            MD5 md;
            string result;
            result = md.pad(message);
            cout << "MD5 result: " << result << endl;
            string b;
            b = buffer;

            if (b == result)
            {
                cout << "true";
            }
            else
            {
```

NKU

```
42                  cout << "flase" << endl;
43                  string finala = htob(b);
44                  string finalb = htob(result);
45                  cout << "hex:" << finala << endl;
46                  cout << "hex:" << finalb << endl;
47                  int count = 0;
48                  for (int i = 0; i < 128; i++)
49                  {
50                      if (finala[i] != finalb[i])
51                          count++;
52                  }
53                  cout << "改变位数:" << count << endl;
54              }
55
56              cout << endl;
57          }
58      }
59
60
61      system("pause");
62      return 0;
63  }
```

五、 实验结果

(一) 样例测试

```
TEST1
MESSAGE IS:
MD5 result: d41d8cd98f00b204e9800998ecf8427e
true
TEST2
MESSAGE IS:  a
MD5 result: 0cc175b9c0f1b6a831c399e269772661
true
TEST3
MESSAGE IS:  abc
MD5 result: 900150983cd24fb0d6963f7d28e17f72
true
TEST4
MESSAGE IS:  message digest
MD5 result: f96b697d7cb7938d525a2f31aaf161d0
true
TEST5
MESSAGE IS:  abcdefghijklmnopqrstuvwxyz
MD5 result: c3fcd3d76192e4007dfb496cca67e13b
true
TEST6
MESSAGE IS:  ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
MD5 result: d174ab98d277d9f5a5611c2c9f419d9f
true
TEST7
MESSAGE IS:  12345678901234567890123456789012345678901234567890123456789012345678901234567890
MD5 result: 57edf4a22be3c955ac49da2e2107b67a
true
```

图 2: 样例测试结果

(二) 雪崩效应检测

```
----------------SNOW TEST--------------------
TEST8
MESSAGE IS:  ibc
MD5 result: 63c3ae7301647113d04e201e02f67b18
flase
hex:10010000000000010101000010011000001111001101001001001111101100001101011010010110001111110111110100101000111000010111
111101110010
hex:01100011110000111010111001110011000000010110010001110001000100111101000001001110001000000001111000000010111101100111
101100011000
改变位数:68

TEST9
MESSAGE IS:  aba
MD5 result: 79af87723dc295f95bdb277a61189a2a
flase
hex:10010000000000010101000010011000001111001101001001001111101100001101011010010110001111110111110100101000111000010111
111101110010
hex:01111001101011111000011101110010001111011100001010010101111110010101101111011011001001110111101001100001000110001001
101000101010
改变位数:61

TEST10
MESSAGE IS:  abg
MD5 result: 894852696ee75656ba33c03041b1fa7f
flase
hex:10010000000000010101000010011000001111001101001001001111101100001101011010010110001111110111110100101000111000010111
111101110010
hex:10001001010010000101001001101001011011101110011101010110010101101011101000110011110000000011000001000001101100011111
101001111111
改变位数:59
```

图 3: 雪崩效应检测

```
TEST11
MESSAGE IS:  abk
MD5 result: 15ef0ee43032ec645b40f84193c045b5
flase
hex:10010000000000010101000010011000001111001101001001001111101100001101011010010110001111110111110100101000111000010111111101100
hex:00010101111011111000011101110010000110000001100101110110001100100010110110100000011111000010000011001001111000000010001011010
改变位数:67

TEST12
MESSAGE IS:  ebc
MD5 result: 01e40bb5c70ff7f7d1e46cc86d40f654
flase
hex:10010000000000010101000010011000001111001101001001001111101100001101011010010110001111110111110100101000111000010111111101100
hex:00000001111001000000101110110101110001110000111111111011111110111110100011110010001101100110010000110110101000000111101100101010
改变位数:66

TEST13
MESSAGE IS:  abC
MD5 result: 36cf1fa7e384dfd385f07a50ffdf0418
flase
hex:10010000000000010101000010011000001111001101001001001111101100001101011010010110001111110111110100101000111000010111111101100
hex:00110110110011110001111110100111111000111000010011011111110100111000010111110000011110100101000011111111101111100000100000110
改变位数:73

TEST14
MESSAGE IS:  aBc
MD5 result: dbbbbe4975e026e04a687871f296a2b2
flase
hex:10010000000000010101000010011000001111001101001001001111101100001101011010010110001111110111110100101000111000010111111101100
hex:11011011101110111011111100100100101110101111000000010011011100000010010100110100001111000011100011111001010010110101000101011000
改变位数:67

TEST15
MESSAGE IS:  Abc
MD5 result: 35593b7ce5020eae3ca68fd5b6f3e031
flase
hex:10010000000000010101000010011000001111001101001001001111101100001101011010010110001111110111110100101000111000010111111101100
hex:00110101010110010011101101111100111001010000010000011101010111000011110010100110100011111101010110110110111100111110000000110
改变位数:59
```

图 4: 雪崩效应检测

平均修改位数为 520/8=65, 而 md5 的 hash 值为 128 位，改变位数约占 50%。所以我们可以认为修改后位数改变几乎占一半。参考文献 [1]

六、 附录

本次实验相关资源已经上传至 github:https://github.com/Metetor/NKU_Cryptography

NKU

参考文献

[1] 吴世忠、宋晓龙、郭涛等译 Paul Garrett 著. *An Introduction to Cryptology*. 机械工业出版社, 2003.

NKU