南开大学

Nankai University

南　开　大　学

网络空间安全学院

密码学课程报告

第四次实验报告

——Hash 函数 MD5

学号：1611519

姓名：周子祎

年级：2016 级

专业：信息安全-法学

2018 年 12 月 22 日

密码学第四次实验报告

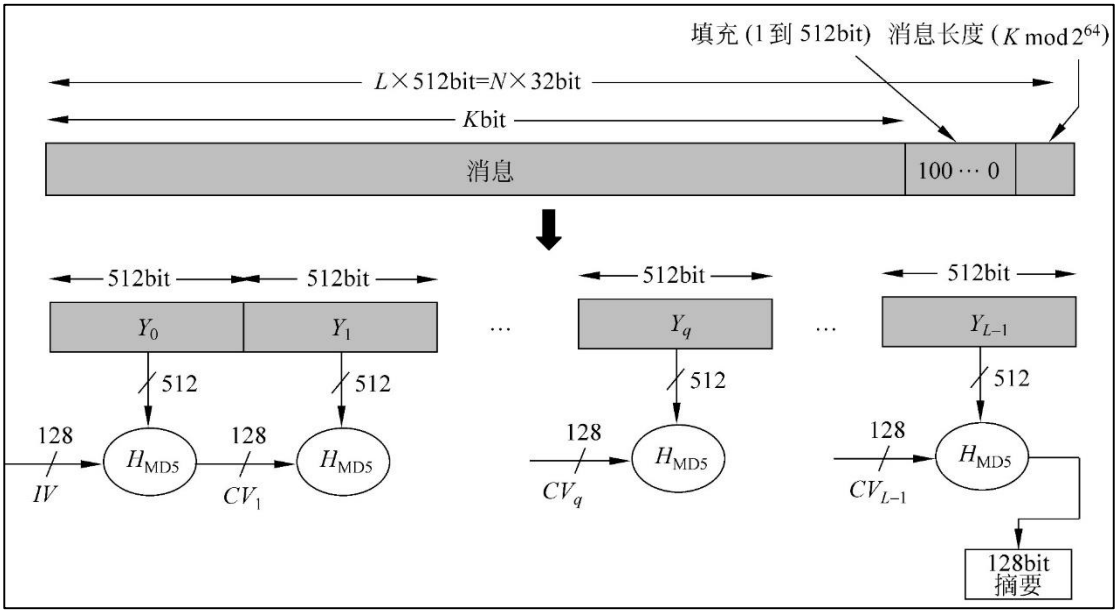
——Hash 函数 MD5

一、 实验目的

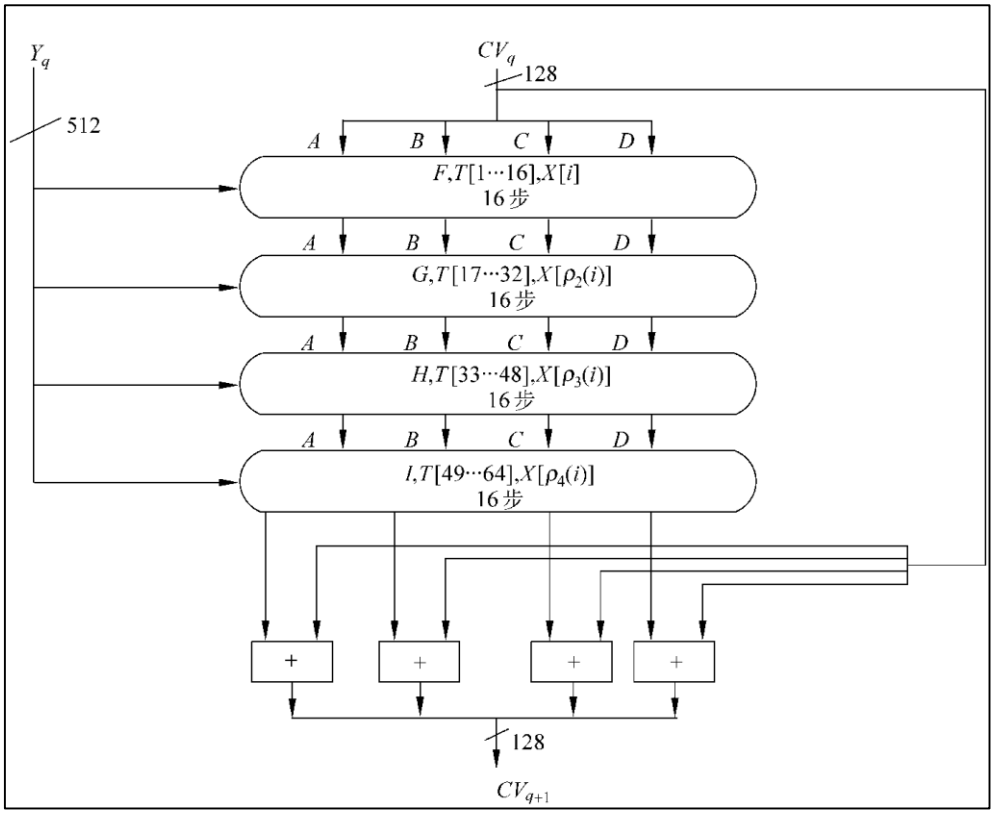
通过实际编程了解 MD5 算法的过程，加深对 Hash 函数的认识。

二、 实验原理

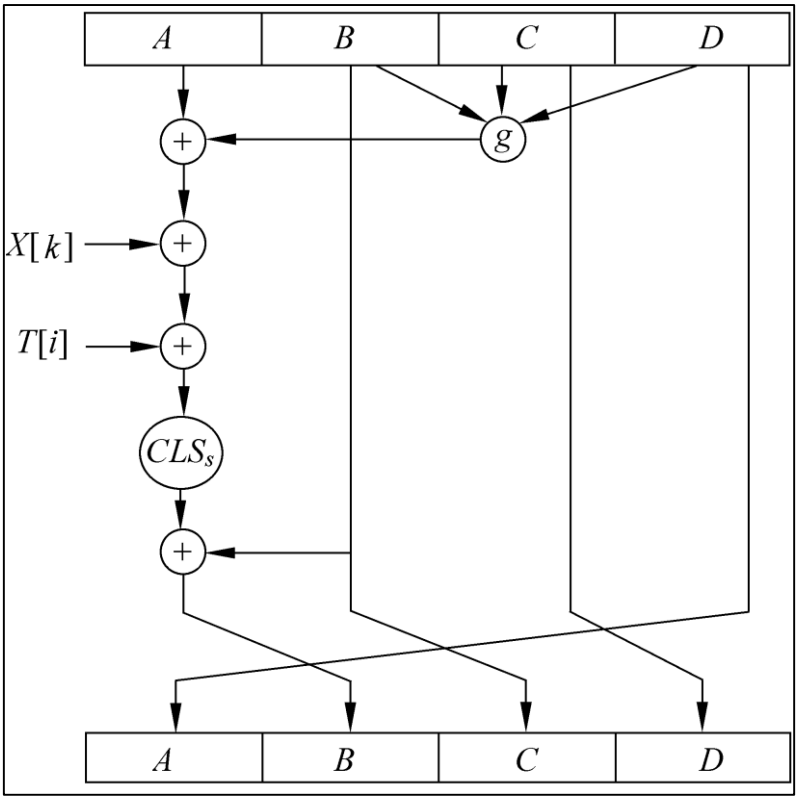
MD5 算法整体框架图如下：



其中对于每个分组进行处理的框架图如下：



对每个分组的处理过程包括四轮压缩，其中每一轮的压缩函数处理过程如下所示：



三、 实验要求

1. 算法实现：

利用 Visual C + + 语言，编写 MD5 的实现代码，并检验代码实现的正确性，提交程序和程序流程图

2. 雪崩效应检验：

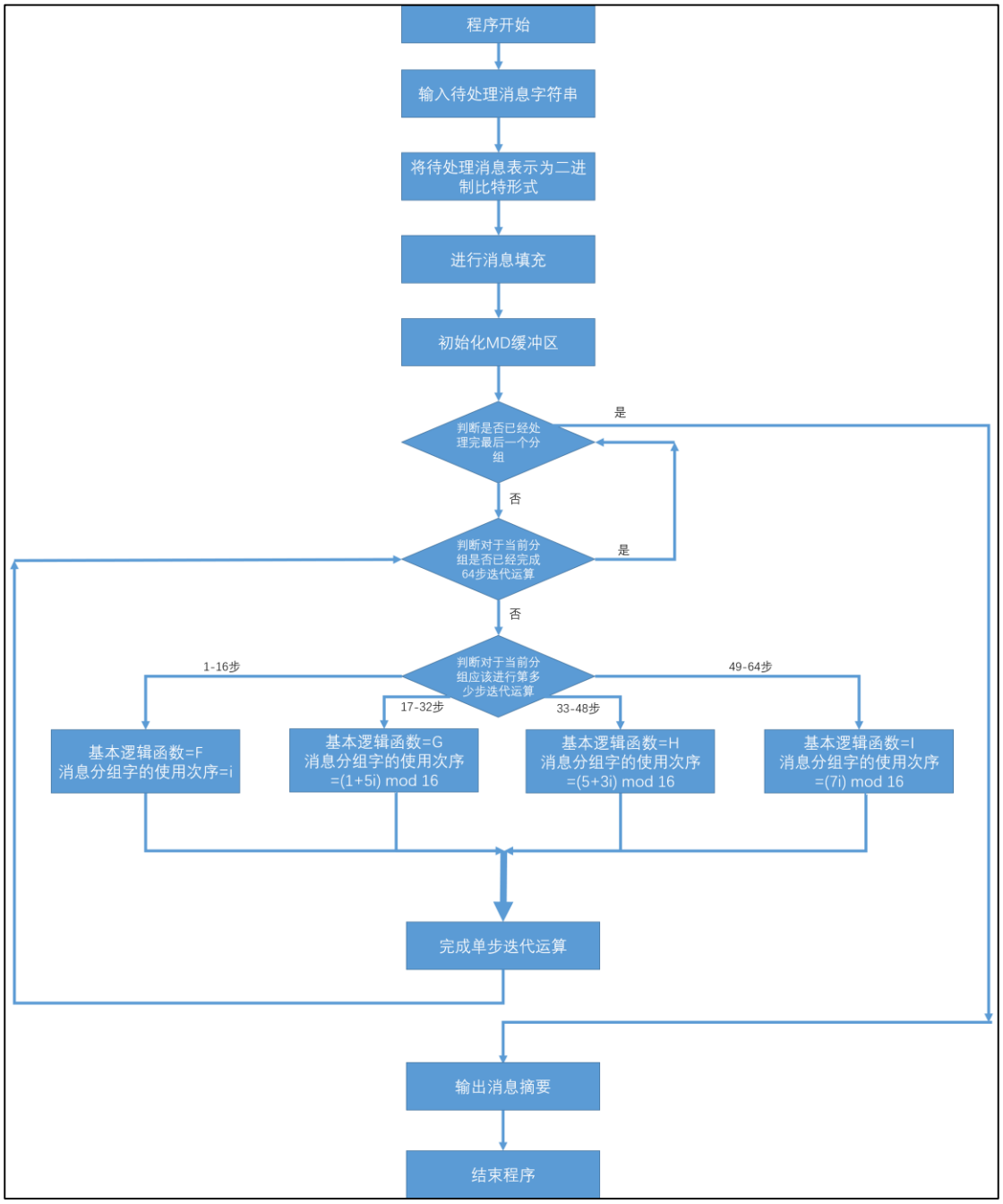
尝试对一个长字符串进行 Hash 运算，并获得其运算结果。对该字符串进行轻微的改动，比如增加一个空格或标点，比较 Hash 结果值的改变位数。进行 8 次这样的测试。

要求给出文本改变前和改变后的 Hash 值，并计算出改变的位数。写出 8 次测试的结果，并计算出平均改变的位数。

四、 实验内容

（一）MD5 算法实现程序

1. MD5 算法程序流程图



2. MD5 算法程序执行结果：

```
**********************************************************************************
*                                MD5哈希算法实现程序                               *
**********************************************************************************
开始测试
----------------------------------------------------------------------------------
请输入消息:
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
----------------------------------------------------------------------------------
128比特摘要 H(Message)为:
d174ab98d277d9f5a5611c2c9f419d9f
----------------------------------------------------------------------------------
请按任意键继续. . .
```

哈希函数结果和文档中给出的哈希函数结果一致:

```
{ "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789",
  { 0xd1, 0x74, 0xab, 0x98, 0xd2, 0x77, 0xd9, 0xf5,
    0xa5, 0x61, 0x1c, 0x2c, 0x9f, 0x41, 0x9d, 0x9f } },
```

3. MD5 算法程序代码:

MD5 核心函数代码如下，详细代码请参见工程文件 hw4_1

```
//核心函数:
//输入: string 类型变量 任意长待处理消息 message
//输出: string 类型变量 128 比特消息摘要
string md5(string message)
{
    //初始化 MD 缓冲区
    unsigned int A = 0x67452301;
    unsigned int B = 0xefcdab89;
    unsigned int C = 0x98badcfe;
    unsigned int D = 0x10325476;

    int lengthInByte = message.length();
    //预留长度，8byte=64bit，用于保存消息被填充前的长度
    //分组长度，64byte=512bit
    //整除：余数部分被忽略
    //+1：如果原始消息长度恰好满足模 512 为 448，也需要进行填充
    int groupNum = ((lengthInByte + 8) / 64) + 1;

    // messageByte 数组中的每一个元素，对应消息二进制表示中的 32bit（4 字节）
    //每个分组的长度为 16 * 32bit = 512bit
    unsigned int *messageByte = new unsigned int[groupNum * 16];

    //清零
    memset(messageByte, 0, sizeof(unsigned int)*groupNum * 16);

    //将 string 类型变量 保存进 messageByte 数组中
    for (int i = 0; i < lengthInByte; i++)
    {
```

```
        //每个字中对应存储 4 个字节的数据
        messageByte[i / 4] |= message[i] << ((i % 4) * 8);
    }

    //进行填充
    messageByte[lengthInByte >> 2] |= 0x80 << ((lengthInByte % 4) * 8);

    //附加信息
    messageByte[groupNum * 16 - 2] = lengthInByte * 8;

    unsigned int a, b, c, d;

    //循环处理每个分组
    for (int i = 0; i < groupNum; i++)
    {
        //转存缓冲区寄存器的值
        a = A;
        b = B;
        c = C;
        d = D;
        unsigned int g;
        int k;

        //对每个分组进行 64 轮压缩
        for (int j = 0; j < 64; j++)
        {
            //根据轮数不同，使用不同的压缩函数
            if ( j < 16 )
            {
                //对 B C D 进行基本逻辑函数运算
                g = F(b, c, d);
                //计算 k，以从消息分组中取出对应的字
                k = j;
            }
            else if ( j>=16 && j < 32 )
            {
                //对 B C D 进行基本逻辑函数运算
                g = G(b, c, d);
                //计算 k，以从消息分组中取出对应的字
                k = (1 + 5 * j) % 16;
            }
            else if ( j>=32 && j < 48 )
            {
                //对 B C D 进行基本逻辑函数运算
                g = H(b, c, d);
                //计算 k，以从消息分组中取出对应的字
                k = (5 + 3 * j) % 16;
            }
            else if ( j>=48 && j < 64 )
            {
                //对 B C D 进行基本逻辑函数运算
                g = I(b, c, d);
                //计算 k，以从消息分组中取出对应的字
                k = (7 * j) % 16;
            }

            unsigned tempd = d;
```

```
            d = c;
            c = b;
            b = b + shift(a + g + messageByte[i * 16 + k] + T[j], s[j]);
            a = tempd;
        }
        A = a + A;
        B = b + B;
        C = c + C;
        D = d + D;
    }
    //返回 16 进制形式字符串
    return Int2HexString(A) + Int2HexString(B) + Int2HexString(C) +
Int2HexString(D);
}
```

其中 Int2HexString 函数用于将 32 位 int 型整数，转化为大端寻址模式，以 16 进制字符串形式输出，代码实现如下：

```
//将 32 位 int 型整数，转化为大端寻址模式，以 16 进制字符串形式输出
//输入：int 类型变量
//输出：string 类型变量
string Int2HexString(int origin)
{
    const char str16[] = "0123456789abcdef";
    unsigned hexNum;

    string temp;
    string hexString = "";

    for (int i = 0; i < 4; i++)
    {
        //对字中的 4 个字节逐一做处理
        temp = "";

        //从 origin 中取对应的 8bit，得到该字节所表示的值
        hexNum = (origin >> (i * 8)) & 0xff;

        for (int j = 0; j < 2; j++)
        {
            //每四位对应一个 16 进制数
            //在 temp 的 0 处插入 1 个 str16[hexNum % 16]
            temp.insert(0, 1, str16[hexNum % 16]);
            hexNum /= 16;
        }

        hexString += temp;
    }
    return hexString;
}
```

（二）MD5 雪崩效应测试

1. MD5 雪崩效应程序执行结果

进行 50 次雪崩效应测试，每次改变消息中的一个字符的一个比特位，测试二进制格式摘要中，发生的比特位的改变位数。

程序执行结果如下：

```
**************************************************************************
*                          MD5哈希算法雪崩效应测试程序                          *
**************************************************************************
开始测试
**************************************************************************
原始消息：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
--------------------------------------------------------------------------
原始摘要：d174ab98d277d9f5a5611c2c9f419d9f
**************************************************************************

**************************************************************************
进行第1次雪崩测试
--------------------------------------------------------------------------
原始消息：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
--------------------------------------------------------------------------
新消息1：@BCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
--------------------------------------------------------------------------
原始摘要：d174ab98d277d9f5a5611c2c9f419d9f
--------------------------------------------------------------------------
新摘要1：457b690b3fafe8809a4a7acc72ee1caf
--------------------------------------------------------------------------
摘要相差二进制位数：65
**************************************************************************

**************************************************************************
进行第2次雪崩测试
--------------------------------------------------------------------------
原始消息：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
--------------------------------------------------------------------------
新消息2：ACCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
--------------------------------------------------------------------------
原始摘要：d174ab98d277d9f5a5611c2c9f419d9f
--------------------------------------------------------------------------
新摘要2：715d406994dbc237a3d03e8badc5fcdd
--------------------------------------------------------------------------
摘要相差二进制位数：53
**************************************************************************
```

```
**********************************************************************************
进行第3次雪崩测试
---------------------------------------------------------------------------------
原始消息：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
---------------------------------------------------------------------------------
新消息3：ABBDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
---------------------------------------------------------------------------------
原始摘要：d174ab98d277d9f5a5611c2c9f419d9f
---------------------------------------------------------------------------------
新摘要3：ca9992f1dcd096f73d637e4f28d18c9f
---------------------------------------------------------------------------------
摘要相差二进制位数：53
**********************************************************************************

**********************************************************************************
进行第4次雪崩测试
---------------------------------------------------------------------------------
原始消息：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
---------------------------------------------------------------------------------
新消息4：ABCEEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
---------------------------------------------------------------------------------
原始摘要：d174ab98d277d9f5a5611c2c9f419d9f
---------------------------------------------------------------------------------
新摘要4：1654d278e8abe922cc3521e2a3cdd80b
---------------------------------------------------------------------------------
摘要相差二进制位数：61
**********************************************************************************

**********************************************************************************
进行第5次雪崩测试
---------------------------------------------------------------------------------
原始消息：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
---------------------------------------------------------------------------------
新消息5：ABCDDFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
---------------------------------------------------------------------------------
原始摘要：d174ab98d277d9f5a5611c2c9f419d9f
---------------------------------------------------------------------------------
新摘要5：f0980d2999123d9121a45c4a052b30e6
---------------------------------------------------------------------------------
摘要相差二进制位数：59
**********************************************************************************
```

```
**********************************************************************************
进行第6次雪崩测试
---------------------------------------------------------------------------------
原始消息：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
---------------------------------------------------------------------------------
新消息6：ABCDEGGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
---------------------------------------------------------------------------------
原始摘要：d174ab98d277d9f5a5611c2c9f419d9f
---------------------------------------------------------------------------------
新摘要6：a49feb3f7109a5e7cee99abc2ef01a20
---------------------------------------------------------------------------------
摘要相差二进制位数：65
**********************************************************************************

**********************************************************************************
进行第7次雪崩测试
---------------------------------------------------------------------------------
原始消息：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
---------------------------------------------------------------------------------
新消息7：ABCDEFFHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
---------------------------------------------------------------------------------
原始摘要：d174ab98d277d9f5a5611c2c9f419d9f
---------------------------------------------------------------------------------
新摘要7：74bc4a66e1e4c5234764ead6875456ae
---------------------------------------------------------------------------------
摘要相差二进制位数：65
**********************************************************************************

**********************************************************************************
进行第8次雪崩测试
---------------------------------------------------------------------------------
原始消息：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
---------------------------------------------------------------------------------
新消息8：ABCDEFGIIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
---------------------------------------------------------------------------------
原始摘要：d174ab98d277d9f5a5611c2c9f419d9f
---------------------------------------------------------------------------------
新摘要8：707f20f5aaa5bd3b3bb4d1ea32b31142
---------------------------------------------------------------------------------
摘要相差二进制位数：69
**********************************************************************************
```

```
**********************************************************************************
进行第48次雪崩测试
----------------------------------------------------------------------------------
原始消息：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
----------------------------------------------------------------------------------
新消息48：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuwwxyz0123456789
----------------------------------------------------------------------------------
原始摘要：d174ab98d277d9f5a5611c2c9f419d9f
----------------------------------------------------------------------------------
新摘要48：5e53694b6dafd8ec3ce8398905625ec6
----------------------------------------------------------------------------------
摘要相差二进制位数：61
**********************************************************************************

**********************************************************************************
进行第49次雪崩测试
----------------------------------------------------------------------------------
原始消息：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
----------------------------------------------------------------------------------
新消息49：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvvxyz0123456789
----------------------------------------------------------------------------------
原始摘要：d174ab98d277d9f5a5611c2c9f419d9f
----------------------------------------------------------------------------------
新摘要49：26be0a953039f09aa001a29da365ca55
----------------------------------------------------------------------------------
摘要相差二进制位数：63
**********************************************************************************

**********************************************************************************
进行第50次雪崩测试
----------------------------------------------------------------------------------
原始消息：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
----------------------------------------------------------------------------------
新消息50：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwyyz0123456789
----------------------------------------------------------------------------------
原始摘要：d174ab98d277d9f5a5611c2c9f419d9f
----------------------------------------------------------------------------------
新摘要50：1d33296f7ef9ad741ea27bab8ff8dbe3
----------------------------------------------------------------------------------
摘要相差二进制位数：64
**********************************************************************************

**********************************************************************************
摘要位数改变情况：
65 53 53 61 59 65 65 69 55 68 74 54 63 63 63 58 65 70 62 69 58 70 61 65 60
65 70 70 57 75 74 74 64 64 68 61 70 59 71 69 62 64 57 63 69 60 79 61 63 64
摘要平均改变位数为：64位
```

2. MD5 雪崩效应测试结果

50 次测试中，消息的二进制改变位数依次为：

65 53 53 61 59 65 65 69 55 68 74 54 63 63 63 58 65 70 62 69 58 70 61 65 60 65

70 70 57 75 74 74 64 64 68 61 70 59 71 69 62 64 57 63 69 60 79 61 63 64

平均改变位数为：64 位

3. MD5 雪崩效应测试代码

MD5 雪崩效应测试主函数代码如下，详细代码请参见工程文件 hw4_2

```
#include<iostream>
#include<string>
#include"md5.h"

using namespace std;


int main()
{
    //存储原始消息
    string OrigiMsg =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";

    //存储原始摘要
    string OrigiSumry;

    //存储 50 次改变后的消息
    string NewMsg [50];

    //存储 50 次改变后的摘要
    string NewSumry [50];

    //存储 50 次改变后的摘要比特变化位数
    int SumryBitChanged [50];

    cout <<
"*************************************************************************
*****" << endl;
    cout << "*                        MD5 哈希算法雪崩效应测试程序
*" << endl;
    cout <<
"*************************************************************************
*****" << endl;
    cout << "开始测试" << endl;
    cout <<
"*************************************************************************
*****" << endl;
    cout << "原始消息: "<< OrigiMsg << endl;
    cout << "-----------------------------------------------------------
-----------------" << endl;
    OrigiSumry = md5(OrigiMsg);
    cout << "原始摘要: "<< OrigiSumry << endl;
    cout <<
"*************************************************************************
*****" << endl;
    cout <<endl;

    for(int i = 0;i<50;i++)
    {
```

```
        cout <<
"**********************************************************************
*****" << endl;
        cout << "进行第"<< i+1<<"次雪崩测试" << endl;
        cout << "----------------------------------------------------------
--------------------" << endl;
        cout << "原始消息："<< OrigiMsg << endl;
        cout << "----------------------------------------------------------
--------------------" << endl;
        NewMsg[i] = OrigiMsg;
        //改变一个字符的一个比特
        //这里采用的方法是：OrigiMsg 的第一个字符，对应的二进制数值，末位为 1
则变为 0；末位为 0 则变为 1
        if(NewMsg[i][i] % 2 ==1)
            NewMsg[i][i]-=1;
        else
            NewMsg[i][i]+=1;

        cout << "新消息"<< i+1<<"： " <<NewMsg[i]<< endl;
        cout << "----------------------------------------------------------
--------------------" << endl;
        cout << "原始摘要："<< OrigiSumry << endl;
        cout << "----------------------------------------------------------
--------------------" << endl;
        NewSumry[i] = md5(NewMsg[i]);
        cout << "新摘要"<< i+1<<"： " <<NewSumry[i]<< endl;
        cout << "----------------------------------------------------------
--------------------" << endl;

        SumryBitChanged[i] = BitCmp(NewSumry[i],OrigiSumry);

        cout << "摘要相差二进制位数："<< SumryBitChanged[i] << endl;
        cout <<
"**********************************************************************
*****" << endl;
        cout << endl;
    }
    cout <<
"**********************************************************************
*****" << endl;
    cout << "摘要位数改变情况："<<endl;
    int ave=0;
    for(int t = 0;t<50;t++)
    {
        ave += SumryBitChanged [t];
        cout<<SumryBitChanged [t] <<" ";
        if(t == 24)
            cout<<endl;
    }
    cout<<endl;

    ave /= 50;
    cout << "摘要平均改变位数为："<< ave << "位"<<endl;

    system("pause");
```

```
    return 0;
}
```

其中 BitCmp 函数用于比较两个 16 进制数对应的 string，所对应的二进制数的比特位差异数，其实现代码如下：

```
int BitCmp(string NewSumry,string OrigiSumry)
{
    int BitDiff = 0;

    //NewSumry、OrigiSumry 均为包含 32 个字符的字符串，每个字符代表一个 16 进制数，4 比特
    //将其分别分为 8 部分，分别进行比较，方便处理
    for(int part=0;part<8;part++)
    {
        //新摘要取出 4 个 char，转为 1 个 long
        string NewSumry_temp =NewSumry.substr(0+4*part, 4);
        char * end_1;
        long NewSumry_Int =
static_cast<long>(strtol(NewSumry_temp.c_str(),&end_1,16));

        //原始摘要取出 4 个 char，转为 1 个 long
        string OrigiSumry_temp =OrigiSumry.substr(0+4*part, 4);
        char * end_2;
        long OrigiSumry_Int =
static_cast<long>(strtol(OrigiSumry_temp.c_str(),&end_2,16));

        //对两个 long 整数，比较其二进制表示中相差的位数
        for(int round = 0;round <16;round++ )
        {
            if( NewSumry_Int%2 != OrigiSumry_Int%2)
            {
                BitDiff++;
            }
            NewSumry_Int /= 2;
            OrigiSumry_Int /= 2;
        }
    }

     return BitDiff;
}
```