网络安全技术

实
验
报
告

学　　院　网安学院

年　　级　大三

班　　级　信息安全

学　　号　**2011762**

姓　　名　于文明

2023 年 **5** 月 **29** 日

目录

一、实验目的

1.深入理解 MD5 算法的基本原理
2.掌握利用 MD5 算法生成数据摘要的所有计算过程
3.掌握 Linux 系统中检测文件完整性的基本方法
4.熟悉 Linux 系统中文件的基本操作方法

二、实验内容

1.准确地实现 MD5 算法的完整计算过程
2.对于任意长度的字符串能够生成 128 位 MD5 摘要
3.对于任意大小的文件能够生成 128 位 MD5 摘要
4.通过检查 MD5 摘要的正确性来检验原文件的完整性

三、实验步骤及实验结果

(一) 实验步骤

1. 配置相关环境和代码框架

实验环境如下:

- 操作系统：ubuntu 18 (vmware)
- 开发工具: vscode
- 编程语言: C++
- 编译配置工具: Cmake

2. 程序文件构成

运行 tree 命令生成文件目录树

```
D:.
 |   CMakeLists.txt
 |   COPYRIGHT
 |   README.md
 |   tree.txt
 |
 ├──.vscode
 |       settings.json
 |
 ├──doc
 |       实验三 （第 5 章 基于 MD5 算法的文件完整性校验程
序.pdf
 |
 ├──include
```

```
│       helperFunc.h
│       md5.h
│
├──src
│       CMakeLists.txt
│       helperFunc.cpp
│       Main.cpp
│       md5.cpp
│
└──test
        nankai.md5
        nankai.txt
```

- Doc 文件夹下存储着实验相关文档材料
- include 文件夹为头文件
- src 文件夹为 cpp 文件
- Test 存储着 md5 正确性验证文件

3. 核心代码

3.1 宏定义

MD5 的实现需要预定义一些运算宏和类型宏定义

```
typedef unsigned char BYTE;
typedef unsigned int DWORD;

#define S11 7
#define S12 12
#define S13 17
#define S14 22
#define S21 5
#define S22 9
#define S23 14
#define S24 20
#define S31 4
#define S32 11
#define S33 16
#define S34 23
#define S41 6
#define S42 10
#define S43 15
#define S44 21

#define F(x,y,z) (((x)&(y))|((~x)&(z)))
#define G(x,y,z) (((x)&(z))|((y)&(~z)))
#define H(x,y,z) ((x)^(y)^(z))
```

```
#define I(x,y,z) ((y)^((x)|(~z)))
#define ROTATE_LEFT(x,n) (((x)<<(n))|((x)>>(32-(n))))
#define FF(a,b,c,d,x,s,ac) { a+=F(b,c,d)+x+ac; a=ROTATE_LEFT(a,s); a+=b;}
#define GG(a,b,c,d,x,s,ac) { a+=G(b,c,d)+x+ac; a=ROTATE_LEFT(a,s); a+=b;}
#define HH(a,b,c,d,x,s,ac) { a+=H(b,c,d)+x+ac; a=ROTATE_LEFT(a,s); a+=b;}
#define II(a,b,c,d,x,s,ac) { a+=I(b,c,d)+x+ac; a=ROTATE_LEFT(a,s); a+=b;}
```

3.2 MD5 类实现

MD5 类定义了核心的摘要计算函数比如 Update 和 Transform，并对各种形式的输入提供了 Update 接口

```
class MD5
{
public:
    MD5();
    MD5(const string &str);
    MD5(ifstream &in);
    //对给定长度的输入流进行 MD5 运算
    void Update(const void* input,size_t length);
    //对给定长度的字符串进行 MD5 运算
    void Update(const string &str);
    void Update(ifstream &in);//对文件中的内容进行 MD5 运算
    const BYTE* GetDigest();
    string Tostring();//将 MD5 摘要以字符串形式输出
    void Reset();
private:
    //对给定长度的字节流进行 MD5 运算
    void Update(const BYTE* input,size_t length);
    void Stop();//用于终止摘要计算过程，输出摘要
    void Transform(const BYTE block[64]);//对消息分组进行 MD5 运算
    //将双字流转换为字节流
    void Encode(const DWORD *input,BYTE *output,size_t length);
    //将字节流转换为双字流
    void Decode(const BYTE *input,DWORD *output,size_t length);
    //将字节流按照十六进制字符串输出
    string BytesToHexString(const BYTE *input,size_t length);
```

```
private:
    DWORD state[4];//用于表示 4 个初始向量
    DWORD count[2];//用于计数，count[0]表示低位，count[1]表示高位
    BYTE buffer_block[64];//用于保存计算过程中按块划分后剩下的比特流
    BYTE digest[16]; //用于保存 128 比特长度的摘要
    bool is_finished;//用于标志摘要计算过程是否结束

    static const BYTE padding[64];//用于保存消息后面填充的数据块
    static const char hex[16];//用于保存 16 进制的字符
};
```

3.3 Update 函数

因为 Update 函数和 transform 函数在材料里面实现的已经比较完整，这里就不再赘述。

3.4 Stop 函数

用于终止摘要计算过程，输出摘要。具体来说是进行补充尾部以及进行最后的运算。首先是计算信息总长度和最后一个分组长度，进而计算出需要补足的长度。第一次调用 Update 函数是补充需要补充的长度并进行一轮运算。第二次调用 Update 函数是补充信息长度到末尾并进行最后一次运算。

```
void MD5::Stop(){
    //printf("Stop Entered\n");
    BYTE bytes[8];
    DWORD oldState[4];
    DWORD oldCount[2];
    DWORD index,padLen;
    memcpy(oldState,state,16);
    memcpy(oldCount,count,8);
    Encode(count,bytes,8);
    index=(DWORD)((count[0]>>3)&0x3f);
    padLen=(index<56)?(56-index):(120-index);
    Update(padding,padLen);
    Update(bytes,8);
    Encode(state,digest,16);

    memcpy(state,oldState,16);
    memcpy(count,oldCount,8);
    //memset(buffer_block,0,sizeof(buffer_block));
    //memset(count,0,sizeof(count));
    is_finished=true;
```

```
}
```

(二) 实验结果

1. 打印帮助信息

```
metetor@metetor-virtual-machine:/mnt/hgfs/网络安全/Lab3/build/bin$ ./MyMD5sum -h
MD5:usage:
        [-h] --help information
        [-t] --test MD5 application
        [-c] [file path of the file computed]
                --compute MD5 of the given file
        [-v] [file path of the file validated]
                --validate the integrality of a given file by manual input MD5 value
        [-f] [file path of the file validated] [file path of the .md5 file]
                --validate the integrality of a given file by read MD5 value from .md5 file
```

2. 打印测试信息

```
metetor@metetor-virtual-machine:/mnt/hgfs/网络安全/Lab3/build/bin$ ./MyMD5sum -t
MD5("") = d41d8cd98f00b204e9800998ecf8427e
MD5("a") = 0cc175b9c0f1b6a831c399e269772661
MD5("abc") = e638f7d51818758264fa897a551e5511
MD5("message digest") = 4c5308173dcd21d555e317698f56f78e
MD5("abcdefghijklmnopqrstuvwxyz") = 873c00d982f204ccdab885861de7bc8e
MD5("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789") = 954c3f7314357d821aba56216d5b45fc
MD5("12345678901234567890123456789012345678901234567890123456789012345678901234567890") = 6fec75d4e7fcd7e966
46b4c7af96bce2
```

3. 为指定文件生成摘要

```
metetor@metetor-virtual-machine:/mnt/hgfs/网络安全/Lab3/build/bin$ ./MyMD5sum -c ../../test/nankai.txt
The MD5 value of file("../../test/nankai.txt") is b1946ac92492d2347c6235b4d2611184
```

4. 验证文件完整性一

```
metetor@metetor-virtual-machine:/mnt/hgfs/网络安全/Lab3/build/bin$ ./MyMD5sum -v ../../test/nankai.txt
Please input the MD5 value of file("../../test/nankai.txt")...
b1946ac92492d2347c6235b4d2611184
The old MD5 value of file("../../test/nankai.txt") you have input is
b1946ac92492d2347c6235b4d2611184
The new MD5 value of file("../../test/nankai.txt") that has computed is
abdb60390ee5dcf80db30b45433551a7
Match Error! The file has been modified!
metetor@metetor-virtual-machine:/mnt/hgfs/网络安全/Lab3/build/bin$ ./MyMD5sum -v ../../test/nankai.txt
Please input the MD5 value of file("../../test/nankai.txt")...
abdb60390ee5dcf80db30b45433551a7
The old MD5 value of file("../../test/nankai.txt") you have input is
abdb60390ee5dcf80db30b45433551a7
The new MD5 value of file("../../test/nankai.txt") that has computed is
abdb60390ee5dcf80db30b45433551a7
OK! The file is integrated
```

5. 验证文件完整性二

```
metetor@metetor-virtual-machine:/mnt/hgfs/网络安全/Lab3/build/bin$ ./MyMD5sum -f ../../test/nankai.txt ../../test/nankai.md5
The old MD5 value of file("../../test/nankai.txt") in ../../test/nankai.md5 is
abdb60390ee5dcf80db30b45433551a7
The new MD5 value of file("../../test/nankai.txt") that has computed is
abdb60390ee5dcf80db30b45433551a7
OK! The file is integrated
```

四、实验遇到的问题及其解决方法

本次实验遇到的最顽固的问题就是程序在 linux 上输出的 md5 值不正确，经过几天的 debug 才发现是 unsigned long 在 windows 和 linux 上面的 size 不同，重新将 DWORD 定义为 unsigned int 后解决了错误

五、实验结论

本次实验完成了一个基于 MD5 Hash 算法校验文件完整性的程序，在最基础的 md5 算法的基础上做出了改进，为各种输入流提供了接口。