计算机网络安全作业 2

SY1206509　张立鑫

题目

请设计和实现一个算法，把一个文件 M 由 A 传输到 B，并保证：

- 文件 M 的完整性
- B 能够认证 M 的发送方
- 文件 M 的完整性、机密性

算法设计思路

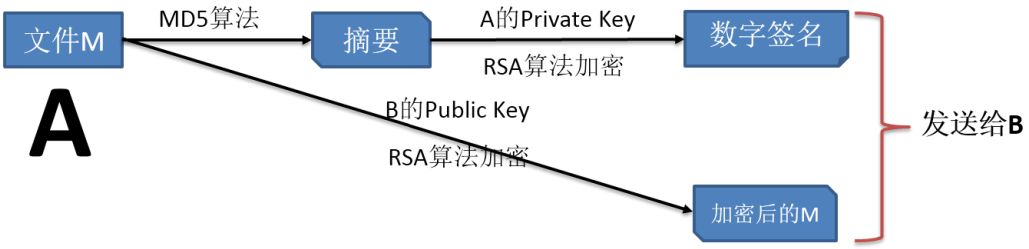
信息完整性和抗否认性是信息安全保证的两个基本要素，数字签名技术通过对消息摘要技术和公开密钥技术的有机结合，实现了对在不可靠网络中传输的信息的完整性和抗否认性的有效保证。

使用数字签名的方式，实现对文件 M 完整性和认证的保证。

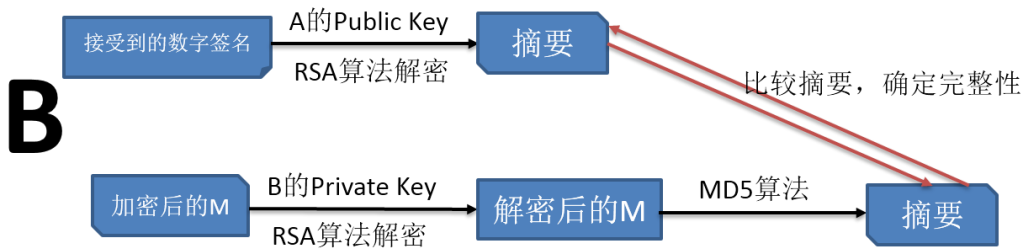
数字签名的大致过程如下：消息发送方首先利用 Hash 函数对待发消息进行摘要处理，生成一个固定长度的消息摘要 H(M)，然后用自己的私钥对该散列值进行加密，形成发送方的一个数字签名，这个数字签名作为消息的附件随消息一起被发送到消息的接受方。接受方利用数字签名中的消息摘要能对消息的完整性进行判断，用发送方的公钥对签名解密则能对发送方进行身份认证和保证抗否认性。

我们使用 MD5 函数作为上述方法的 Hash 函数，使用 RSA 算法作为非对称加密算法。具体实现流程如下：

发送端 A 的操作流程：



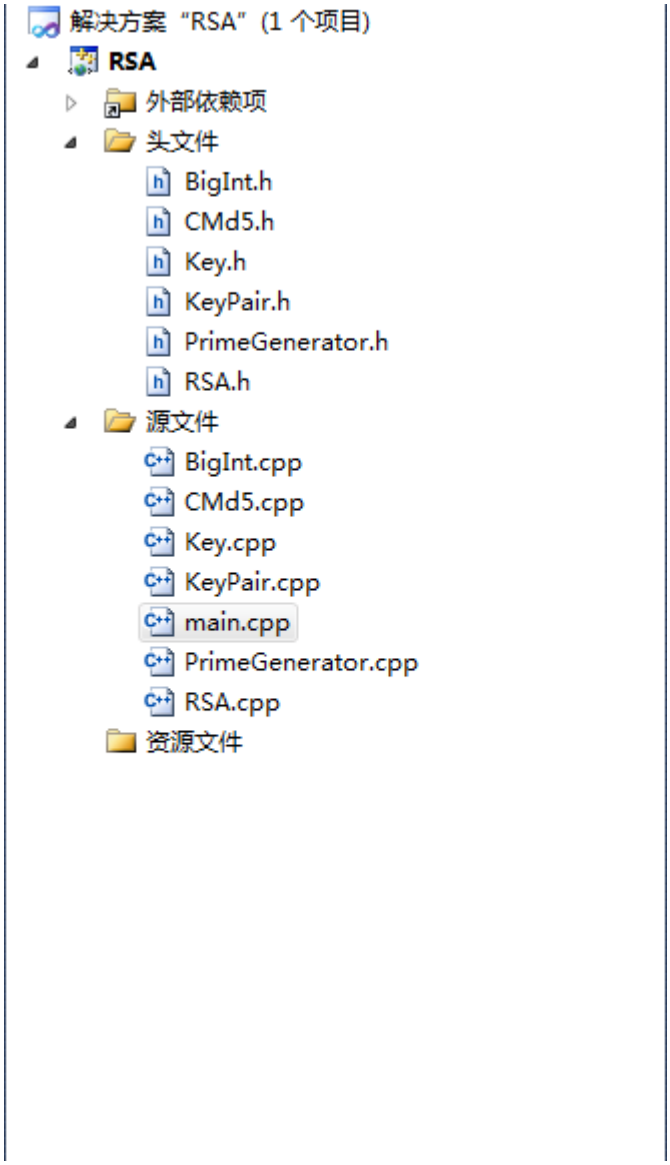
接收端 B 的操作流程：



由上面两图可以清楚知道该算法的工作流程。通过对文件 M 进行数字签名可以保证文件 M 传输后的完整性和抗否认性。对文件 M 使用 B 的 public key 进行加密，使得只有 B 能

够通过自己的 private key 进行解密，保证了文件 M 的机密性。

文件清单

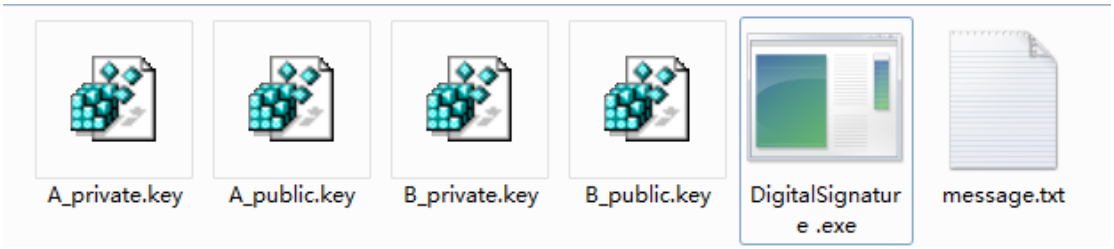


运行方法及相关说明:

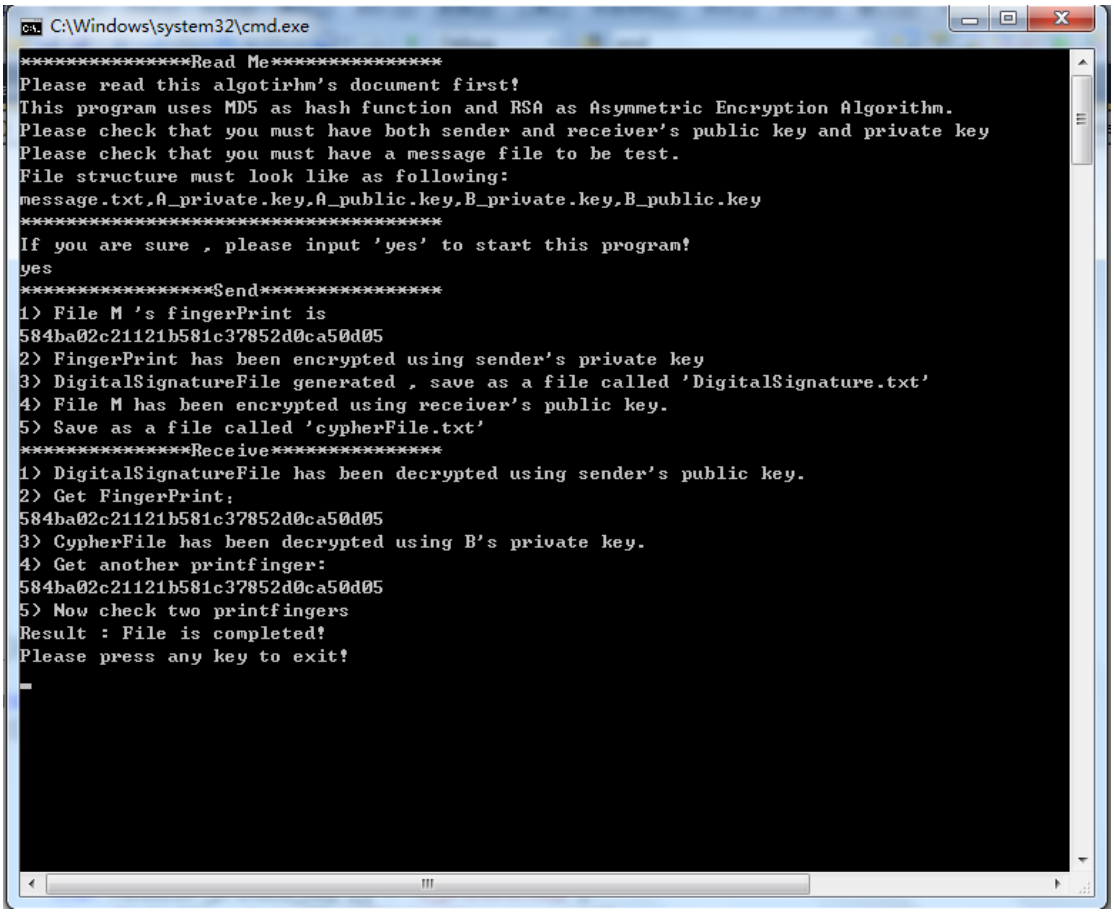
1、 程序采用 c++语言编写，文件清单里的 main.cpp 是主控程序，其他文件问逻辑处理函数或是工具类。

2、 文件 PrimeGenerator 用于生成 public key 与 private key。

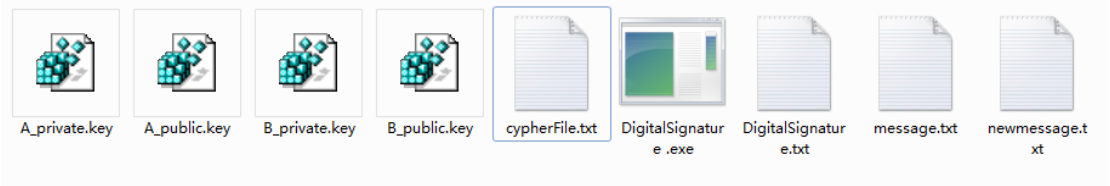
3、 程序基于文件驱动运行，即将各种文件作为参数作为程序的入口。在程序的根目录下必须保证收发端的 public key 以及 private key，以及待发送处理的文件 M。命名规则如下图所示：



4、 运行 DigitalSignature.exe 可执行程序，运行终端显示如下（下图显示了整个工作流程信息，模拟了 send 和 Receive 的过程）：



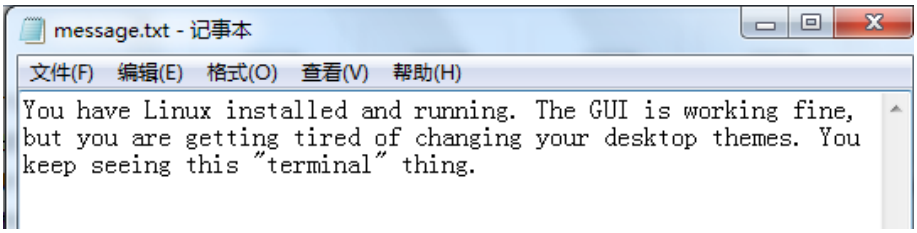
5、 文件输出为



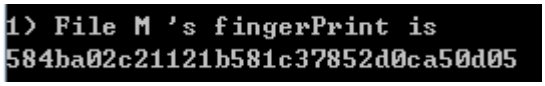
DigitalSignature.txt 表示发送端生成的数字签名，cypherFile.txt 表示使用接受端的 publickey B_public.key 加密 message.txt 后的密文文件。Newmessage 文件表示的是接受端 B 时候自己的 private key 解密后的文件。

6、 一个具体事例的演示：

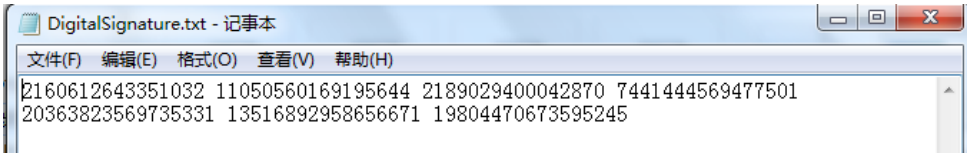
a) message.txt 文件内容



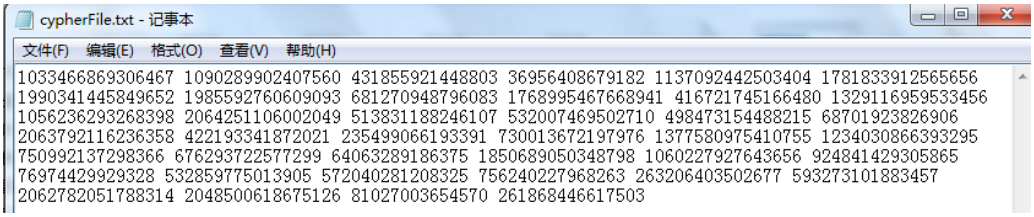
b) 发送端使用 MD5 生成 message.txt 文件的指纹为



c) 发送端使用 A_private.key 加密该指纹得数字签名 DigitalSignature.txt

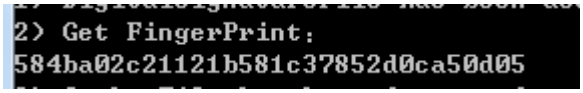


d) 发送端使用 B_public.key 加密文件 message.txt ，并将密文保存为 cypherFile.txt

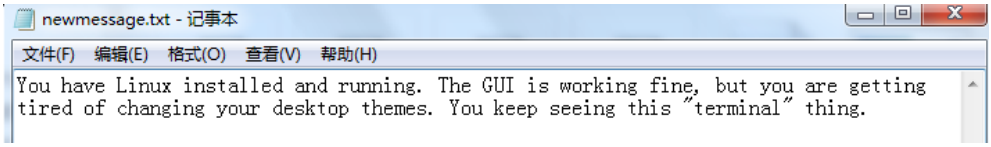


e) 发送端操作结束

f) 接收端使用 A_publickey 解密 DigitalSignature.txt，得到解密后指纹 1

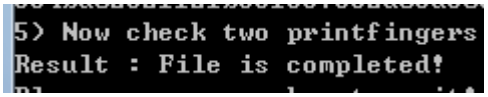


g) 接收端使用 B_privatekey 解密 cypherFile.txt，得到解密后的文件，并写入 newmessage.txt。发现与原文件内容相同。



h) 接收端使用 MD5 生成文件 newmessage.txt 的指纹 2

i) 比较指纹 1 和指纹 2，判断是否可以相同



程序关键代码

本实验采用了网络上开源的 MD5 算法以及 RSA 算法，并对其算法中的一些函数进行了改写

与扩充。先将本实验的核心主程序代码给出，代码附有比较详细的注释信息，这里便不再赘述。

```
#include <cstdlib> //srand()
#include <iostream>//cout
#include <ctime>   //time()
#include <cstring> //strcmp()
#include "test.h"  //testing functions
#include "RSA.h"   //GenerateKeyPair()
#include "PrimeGenerator.h"    //Generate()
#include "CMd5.h"

using std::cout;
using std::endl;
using std::cin;
using namespace std;
/*read a key from a key file*/
Key ReadKey(const char * keyfile)
{
    std::ifstream source(keyfile, std::ios::in | std::ios::binary);
    if (!source){
        cout<< "Error : Opening file \"KeyFile\" failed."<<endl;
    }
    string str_modulus , str_exponent;
    source >> str_modulus >> str_exponent;
    BigInt modulus(str_modulus);
    BigInt exponent(str_exponent);
    Key key(modulus , exponent);
    return key;
}

/*read whole text from a file*/
std::string GetWholeTextFromFile(const char * filename)
{
    std::ifstream source(filename , std::ios::in | std::ios::binary);
    if(!source){
        cout<< "Error : Opening file \"File\" failed."<<endl;
    }
    source.seekg(0, std::ios::end);
    const unsigned long int fileSize = source.tellg();
    source.seekg(0, std::ios::beg);
    char buff[10240];
    source.read(buff , fileSize);
    buff[fileSize] = '\0';
    std::string wholetext = std::string(buff);
```

```
    source.close();
    return wholetext;
}

/*generate a fingerprint using MD5 algorithm*/
std::string GenerateFingerPrintFromFile(const char * filename)
{
    std::ifstream source(filename , std::ios::in | std::ios::binary);
    if(!source){
        cout<< "Error : Opening file \"File\" failed."<<endl;
    }
    source.seekg(0, std::ios::end);
    const unsigned long int fileSize = source.tellg();
    source.seekg(0, std::ios::beg);
    char buff[10240];
    source.read(buff , fileSize);
    buff[fileSize] = '\0';

    unsigned char digest[16];
    //调Ì¡Â用®?MD5相¨¤关?函¡¥数º¨y，ê?生¦¨²成¨¦buff的Ì?MD5码?，ê?存ä?入¨?digest
    md5_state_t md5state;
    md5_init(&md5state);
    md5_append(&md5state, (const unsigned char *)buff, strlen(buff));
    md5_finish(&md5state, digest);
    char presentation[33];
    md5_presentation(digest , presentation);
    presentation[32] = '\0';
    std::string fingerPrint = std::string(presentation);
    source.close();
    return fingerPrint;
}

/*
    Send a messageFile M to a receiver.
    Pass messageFile'path , sender's private key path , receiver's public key
path to the function.
    This function will write M's digitalSignature to a file called
'DigitalSignature.txt';
    Also write encrypted M a file called 'cypherFile.txt' using receiver's public
key.
*/
void send(char * messageFile , char * sender_private_key , char *
receiver_public_key)
{
```

```
    char cypherFile [] = "cypherFile.txt";//output file name
    char digitalSignatureFile [] = "DigitalSignature.txt";
    std:string fingerPrint = GenerateFingerPrintFromFile(messageFile);
    cout<<"1) File M 's fingerPrint is"<<endl<<fingerPrint<<endl;
    Key B_publickey = ReadKey(receiver_public_key);
    Key A_privatekey = ReadKey(sender_private_key);
    std::string cypherText = RSA::Encrypt(fingerPrint, A_privatekey);
    cout<<"2) FingerPrint has been encrypted using sender's private key"<<endl
        <<"3) DigitalSignatureFile generated , save as a file called
'DigitalSignature.txt'"<<endl;
    RSA::Encrypt(messageFile, cypherFile, B_publickey);
    cout<<"4) File M has been encrypted using receiver's public key."<<endl
         <<"5) Save as a file called 'cypherFile.txt'"<<endl;
    std::ofstream dest(digitalSignatureFile, std::ios::out |
std::ios::binary);
    dest<<cypherText;
    dest.flush();
    dest.close();
}


/*
    Receive a encrypted file cypherFile and a digitalSignature file From sender ,
    check whether M is completed or not , decrypt from cypherFile using receiver's
private key.
    Pass paths of cypherFile ,
digitalSignatureFile ,sender_public_key ,receiver_private_key to this
function.
    This function will print some checking results.
*/
void receive(char * cypherFile , char * digitalSignatureFile , char *
sender_public_key , char * receiver_private_key)
{
    char newmessage [] = "newmessage.txt"; // output file name
    Key A_publickey = ReadKey(sender_public_key);
    Key B_privatekey = ReadKey(receiver_private_key);
    std::string cypherText = GetWholeTextFromFile(digitalSignatureFile);
    cout<<"1) DigitalSignatureFile has been decrypted using sender's public
key."<<endl
        <<"2) Get FingerPrint: êo"<<endl;
    std::string newFingerPrint = RSA::Decrypt(cypherText, A_publickey);
    cout<<newFingerPrint<<endl;
    RSA::Decrypt(cypherFile, newmessage, B_privatekey);
    cout<<"3) CypherFile has been decrypted using B's private key."<<endl
```

```
        <<"4) Get another printfinger:"<<endl;
    std::string anotherFingerPrint = GenerateFingerPrintFromFile(newmessage);
    cout<<anotherFingerPrint<<endl;
    cout<<"5) Now check two printfingers"<<endl;
    bool check = (newFingerPrint==anotherFingerPrint);
    if (check == true)
    {
        cout<<"Result : File is completed!"<<endl;
    }
    else
    {
        cout<<"Result : File is not completed,Please check!"<<endl;
    }
}

int main(int argc, char *argv[])
{
    char messageFile [] = "message.txt";
    char sender_private_key [] = "A_private.key";
    char receiver_public_key [] = "B_public.key";

    char sender_public_key [] = "A_public.key";
    char receiver_private_key [] = "B_private.key";
    char digitalSignature [] = "DigitalSignature.txt";
    char cypherFile [] = "cypherFile.txt";

    cout<<"***************Read Me***************"<<endl;
    cout<<"Please read this algotirhm's document first!"<<endl;
    cout<<"This program uses MD5 as hash function and RSA as Asymmetric
Encryption Algorithm."<<endl;
    cout<<"Please check that you must have both sender and receiver's public
key and private key"<<endl;
    cout<<"Please check that you must have a message file to be test."<<endl;
    cout<<"File structure must look like as following:"<<endl;
    cout<<"message.txt,A_private.key,A_public.key,B_private.key,B_public.ke
y"<<endl;
    cout<<"************************************"<<endl;
    std::string input;
    while(1)
    {
        cout<<"If you are sure , please input 'yes' to start this
program!"<<endl;
        cin >> input;
        if (input != "yes")
```

```
            continue;
        cout<<"****************Send***************"<<endl;
        send(messageFile , sender_private_key , receiver_public_key);
        cout<<"***************Receive***************"<<endl;
        receive(cypherFile ,digitalSignature , sender_public_key ,
receiver_private_key);
        break;
    }
    cout<<"Please press any key to exit!"<<endl;
    cin >> input;
    return 0;
}
```