

# 密码学 大作业报告

宣恩允

1911501

2022 年 6 月 14 日

# 目录

<b>1 题目简述</b>	<b>3</b>
<b>2 实现思路</b>	<b>3</b>
2.1 cs.cpp 代码结构 . . . . .	4
2.2 消息结构的定义 . . . . .	5
2.3 server 主程序 . . . . .	6
2.4 发送和接受的线程函数 . . . . .	7
<b>3 运行结果</b>	<b>8</b>

## 1 题目简述

在保密通信过程中，对消息完整性的检验和对消息来源的认证，是一个非常重要的问题，本次大作业的内容为：设计一个协议，利用 rsa 公钥加密算法和 MD5 哈希算法，实现对消息的完整性检验和发送者身份验证的功能，并编程实现这个协议。程序的要求是假设通讯双方为 A 合 B，并假设发方拥有自己的 RSA 公钥 PKA 和私钥 SKA，同时收方 B 已经通过某种方式知道了发方的公钥 PKA。协议要求对发方 A 发来的消息，收方 B 通过检验，能够确定：

1. B 收到的消息是完整的，即消息在传送过程中没有遭到非法修改；
2. B 收到的消息来源是真实的，即该消息的确是由 A 发来的，而不是由其他人伪造的。

## 2 实现思路

实验中需要完成的两个功能是：确保接受的消息的完成性和确保消息来源的可靠性。对于这两个功能，我分别使用了 MD5 和 RSA 加密算法来实现，具体思路如下：

- **确保消息完整性**

这一部分利用 MD5 加密算法，利用其无碰撞性，对需要发送给 B 的消息计算其 MD5 的哈希值，在 B 接受到该条消息时，B 再进行一次 MD5 哈希值的计算，若该次计算的结果与原先的哈希值一致，则可以说明消息在传输的过程中没有被恶意修改。此假设是基于 MD5 的无碰撞性，即两个不同的消息值不会产生相同的 MD5 哈希值。尽管 MD5 算法已经被证明可以在较短时间内找到冲突，但是在本次实验中不考虑可能会冲突的情况。

- **确保消息可靠性**

这部分利用了 RSA 加密算法，对将要发送的信息在末尾加上发送方的签名，例如，在消息后加上 “i’m Alice.”，然后将该加上了签名的信息使用私钥进行加密，再发送给接收方 B。B 收到消息后，由于题目中假设 B 已经通过某种途径获取了发送方的公钥，B 使用该公钥解密接收到的消息，若可以看到消息末尾的附加信息是 “i’m Alice.” 则可以确定该信息是由 Alice 发出的，即信息是可靠的。

在此前的实验中，已经分别完成了 RSA 加密算法和 MD5 加密算法，因此本次实验中，把这两次实验的代码直接当作头文件引入。

实验包含的文件有：

```
1 大作业
2      chat.exe      // 可执行程序
3      Makefile      // 编译文件
4      md5.cpp        // MD5加密算法
5      md5.h
6      rsa.cpp        // RSA加密算法
7      rsa.h
8      sc.cpp         // 主程序
9
10     BigInt
11         binaryHelpFunc.cpp      // 大数类
12         binaryHelpFunc.h
```

由于 MD5 和 RSA 文件沿用前两次实验的代码，在此处不作讲解。这次主要编写的程序是 sc.cpp。

## 2.1 cs.cpp 代码结构

作为本次实验主要编写的部分，其代码结构如下：

```

1 // 开头是一些头文件链接、宏定义以及用到的全局变量
2
3 struct Message...          // 消息结构体
4
5 void initializaion()...     // socket初始化
6
7 void serverRecv(void *param)...      // 服务端接受线程
8
9 void serverSend(void *param)...      // 服务器发送线程
10
11 void clientRecv(void *param)...      // 客户端接收线程
12
13 void clientSend(void *param)...      // 客户端发送线程
14
15 void server()...            // 服务端主程序
16
17 void client()...           // 客户端主程序
18
19 int main()...              // 主函数

```

## 2.2 消息结构的定义

实验中发送时不仅仅只需要发送加密后的信息，还需要附带发送信息的 MD5 摘要和发送方加密后的签名。因此定义了如下的消息结构，在发送时整体发送，整体接受。

```

1 struct Message
2 {
3     char msg[N_LENGTH + 1];

```

```

4     char md5text[33];
5     char signature[N_LENGTH + 1];
6 };

```

MD5 算法会生成 128 位的消息摘要，因此存储 MD5 的数据容量设置为 33（多一位结束符号，不然会出错，这个 bug 也是调试了好几天才发现，结果仅仅是把 32 改成 33 就解决了，以后一定要注意）。RSA 加密算法最多产生 1024 位的加密结果，因此将宏“N\_LENGTH”置为 1024。

## 2.3 server 主程序

initializaion 函数用于初始化 socket 的相关参数，该部分比较简单，在此不予解释。

下面分析 server 向 client 发送消息的实现过程，反过来 client 向 server 发送消息的相同的实现原理。

服务端主程序 server 中，先生成了客户端使用的 RSA 密钥，然后与设置 socket 为监听状态等待连接。在开始聊天前，需要先将服务器的 RSA 公钥密钥发送给已经连接的客户端：

```

1  memset(server_Pk, 0, KEY_LENGTH);
2  strcpy(server_Pk, server_rsa.getPk().getbits().c_str());
3  send(remoteSocket, server_Pk, KEY_LENGTH, 0);

```

在客户端接受：

```

1  memset(server_Pk, 0, KEY_LENGTH);
2  recv(clientSocket, server_Pk, KEY_LENGTH, 0);
3  BigInt a(server_Pk);
4  server_rsa.setPk(server_Pk);

```

发送完毕后需要输入服务器的签名，用于消息传输过程中的身份认证，此后，可以开始进入消息传输。

使用了 C++ 多线程来实现此部分，这样可以实现全双工通信，否则仅是半双工通信。

```
1 __beginthread(serverSend, 0, &remoteSocket);  
2 __beginthread(serverRecv, 0, &remoteSocket);
```

## 2.4 发送和接受的线程函数

- 发送函数

对输入的信息以及签名使用服务器的 RSA 私钥加密，计算加密后的消息的 MD5 信息摘要，包装成结构体后，一起发送给客户端。

```
1     Message sendmsg;  
2     memset(&sendmsg, 0, sizeof(Message));  
3  
4     string temp;  
5  
6     getline(cin, temp);  
7     BigInt a = server_rsa.ency(temp);  
8  
9     // 添加RSA加密后的密文  
10    temp = a.getbits();  
11    strcpy(sendmsg.msg, temp.c_str());  
12  
13    // 添加签名  
14    BigInt b = server_rsa.ency(signature);  
15    temp = b.getbits();  
16    strcpy(sendmsg.signature, temp.c_str());  
17  
18    // 添加MD5摘要
```

```

19         md5.Update(sendmsg.msg);
20         strcpy(sendmsg.md5text, md5.ToString().c_str());
21
22         int len = sizeof(Message);
23         send(remoteSocket, (char *)&sendmsg, len, 0);

```

### • 接收函数

对接受的信息以及签名使用客户端的 RSA 公钥解密，计算解密前消息的 MD5 信息摘要，对比其是否与接受的 MD5 一致，若一致则正确接受，若不一致，则说明传输过程中，消息被篡改。签名解密后，检验是否是服务器的签名，若不是，则说明该信息是伪造的，并不出自服务器。

```

1     Message recvmsg;
2     memset(&recvmsg, 0, sizeof(Message));
3     int len = sizeof(Message);
4     recv(remoteSocket, (char *)&recvmsg, len, 0);

```

## 3 运行结果

终端中输入“make run”运行程序：



```
PS C:\Users\32455\Desktop\zey\南开大学\密码学\大作业> make run
./chat
input "s" to start server, "c" to start client
:
s
Generating RSA keys, please wait...
Generated!
I'm listening...
Successfully connected.
server public key is : 10001
server N is: 826152E78A6C10DDA5D9EF2E774739EE4
547AC3FCE3F6704075957DE95E0635599B6D6316434321
C4111038B2D9B1BB9B727B7402329D1612659DB7B83B6F
8A9
received client public key: 10001
receive client N: 895C327601D2C8EFA1E5391C2EFB
D1A320A9501B860B663D9BA25135958B0391083A709359
06AEA08533E9139136BEB616BE773858ECDFC5EBCC9787
7536F7E1
input your signature:
```

```
PS C:\Users\32455\Desktop\zey\南开大学\密码学\大作业> make run
./chat
input "s" to start server, "c" to start client
:
c
Generating RSA keys, please wait...
Generated!
Successfully connected.
received server public key: 10001
receive server N: 826152E78A6C10DDA5D9EF2E7747
39EE4547AC3FCE3F6704075957DE95E0635599B6D63164
34321C4111038B2D9B1BB9B727B7402329D1612659DB7B
83B6F8A9
client public key is : 10001
client N is: 895C327601D2C8EFA1E5391C2EFBD1A32
0A9501B860B663D9BA25135958B0391083A70935906AEA
08533E9139136BEB616BE773858ECDFC5EBCC97877536F
7E1
input your signature:
```

可以看到双方先将 RSA 密钥发送，然后输入双方的签名：

```
PS C:\Users\32455\Desktop\zey\南开大学\密码学\大作业> make run
./chat
input "s" to start server, "c" to start client
:
s
Generating RSA keys, please wait...
Generated!
I'm listening...
Successfully connected.
server public key is : 10001
server N is: 826152E78A6C10DDA5D9EF2E774739EE4
547AC3FCE3F6704075957DE95E0635599B6D6316434321
C4111038B2D9B1BB9B727B7402329D1612659DB7B83B6F
8A9
received client public key: 10001
receive client N: 895C327601D2C8EFA1E5391C2EFB
D1A320A9501B860B663D9BA25135958B0391083A709359
06AEA08533E9139136BEB616BE773858ECDFC5EBCC9787
7536F7E1
input your signature:
1111
----- <begin chatting> -----
```

```
PS C:\Users\32455\Desktop\zey\南开大学\密码学\大作业> make run
./chat
input "s" to start server, "c" to start client
:
c
Generating RSA keys, please wait...
Generated!
Successfully connected.
received server public key: 10001
receive server N: 826152E78A6C10DDA5D9EF2E7747
39EE4547AC3FCE3F6704075957DE95E0635599B6D63164
34321C4111038B2D9B1BB9B727B7402329D1612659DB7B
83B6F8A9
client public key is : 10001
client N is: 895C327601D2C8EFA1E5391C2EFBD1A32
0A9501B860B663D9BA25135958B0391083A70935906AEA
08533E9139136BEB616BE773858ECDFC5EBCC97877536F
7E1
input your signature:
1100
----- <begin chatting> -----
```

这里服务器签名为“1111”，客户端签名为“1100”，接下来开始发送消息：

```
Windows PowerShell
receive client N: 895C327601D2C8EFA1E5391C2EFB
D1A320A9501B860B663D9BA25135958B0391083A709359
06AEA08533E9139136BEB616BE773858ECDFC5EBCC9787
7536F7E1
input your signature:
1111
----- <begin chatting> -----
1101011
[server]: 71AA3D70F6E048F0D406B77A7A8E736EA503
21D75D082935FABE69807445D584EDAA8C1D7E58AF2161
2CA9130C38895385050BC32FEB473FA5357F0E53FDBD5F
[server]signature: D3CA21BC23A56B3DC34476FF8FA
E3F1B76E5A33E73F392D4A94AAC386DC60487CBE3701A5
920F653C41875888AFCFCB094091563D62EFA9ED44E47D
128B9EDA
[server]md5: 1642dc1c90328694e4cf2b952067b3f0
receive correctly.
[client]: 2C03BA9A9761668F917E7ED529287E9B6F2E
E764D26ABDD3F9A522B06F05D8AF9ACC361085A863F361
AA61621FA7E1FA165BB92FE93E07661AEFFA63BA67A7DE
[client]rsa after decry: 1000011
[client]md5: 3ab5c4299bbac385f14fedd127708ec5
[client]signature: 1F9D8D95E0301CD1C86E4BDA26F
E0471AAF383E55C669A9A44468DEC82258DC76D52ECB36
486AA85C6B7E0C381810EC983F1864FAF00C65A866A7DE
ECD166BC
[client]signature after decry: 1100

client N is: 895C327601D2C8EFA1E5391C2EFBD1A32
0A9501B860B663D9BA25135958B0391083A70935906AEA
08533E9139136BEB616BE773858ECDFC5EBCC97877536F
7E1
input your signature:
1100
----- <begin chatting> -----
receive correctly.
[server]: 71AA3D70F6E048F0D406B77A7A8E736EA503
21D75D082935FABE69807445D584EDAA8C1D7E58AF2161
2CA9130C38895385050BC32FEB473FA5357F0E53FDBD5F
[server]rsa after decry: 1101011
[server]md5: 1642dc1c90328694e4cf2b952067b3f0
[server]signature: D3CA21BC23A56B3DC34476FF8FA
E3F1B76E5A33E73F392D4A94AAC386DC60487CBE3701A5
920F653C41875888AFCFCB094091563D62EFA9ED44E47D
128B9EDA
[server]signature after decry: 1111

1000011
[client]: 2C03BA9A9761668F917E7ED529287E9B6F2E
E764D26ABDD3F9A522B06F05D8AF9ACC361085A863F361
AA61621FA7E1FA165BB92FE93E07661AEFFA63BA67A7DE
[client]signature: 1F9D8D95E0301CD1C86E4BDA26F
E0471AAF383E55C669A9A44468DEC82258DC76D52ECB36
486AA85C6B7E0C381810EC983F1864FAF00C65A866A7DE
ECD166BC
[client]md5: 3ab5c4299bbac385f14fedd127708ec5
```

可以看到双方发送的原始信息，以及加密的信息，以及发送后解密的信息。