技术报告

项 目 名 称 ：主机之间的密钥分发和加密传输

作者 姓 名 ：郑旋

大作业时间 ：2020.7.7—2020.7.15

摘 要

随着信息技术的不断发展，网络安全问题变得愈发突出。在平常的生活中，我们会遇到许多关乎网络安全的问题，例如病毒攻击、信息的盗取等，这使得网络安全成为了一个不可忽视的问题。网络安全是一个关系国家安全和主权、社会的稳定、民族文化的继承和发扬的重要问题。其重要性,正随着全球信息化步伐的加快而变到越来越重要。其中加密通信又是其中一个基本且重要的问题。

该课程设计利用 RSA 算法及 AES 算法，实现了实现两个主机之间的密钥分发和加密传输。其中 RSA 算法用于实现两个主机之间的密钥分发，而 AES 算法实现实现两个主机之间的加密数据传输。主机与主机之间的信息传递必须在网络上完成。

目录

1 总体概述

1.1 研究背景

随着信息技术的不断发展，网络安全的问题愈发严重，如何在网络中实现加密通信变得尤为重要，数据信息的保密成为亟待解决的一个问题。本文给基于基础的加密通信，给出了一种使用使用 AES 加密算法为对称密钥，RSA 加密算法为非对称密钥的加密方法，并给出了具体实现。

1.2 项目目标

本项目旨在能实现两台主机之间的加密通信。这种加密通信是迅速，并具有完整性、机密性、可用性等特征。

2 需求分析

本程序要求能在两台终端之间实现加密信息的互传。设有 A、B 两台主机。其中一台应作为服务器，假设为 A；另一台应作为用户机，假设为 B。

2.1 功能需求

B 主机应能通过 RSA 算法生成公钥私钥对，并通过网络通信将公钥传送到 A 机上。A 机需要分发的对称密钥为“12345678”，因此 A 机使用得到的公钥加密该对称密钥，并用该对称密钥加密待传输的明文信息“NPU”。最终得到由公钥加密后的对称密钥以及由对称密钥加密后的密文。A 机将这两个信息打包发给 B 机，B 机用本机上所拥有的私钥对公钥加密后的对称密钥进行解析，得到对称密钥；然后用解析出来的对称密钥对密文进行解密，得到最终的明文。该过程如图所示 2-1 所示。

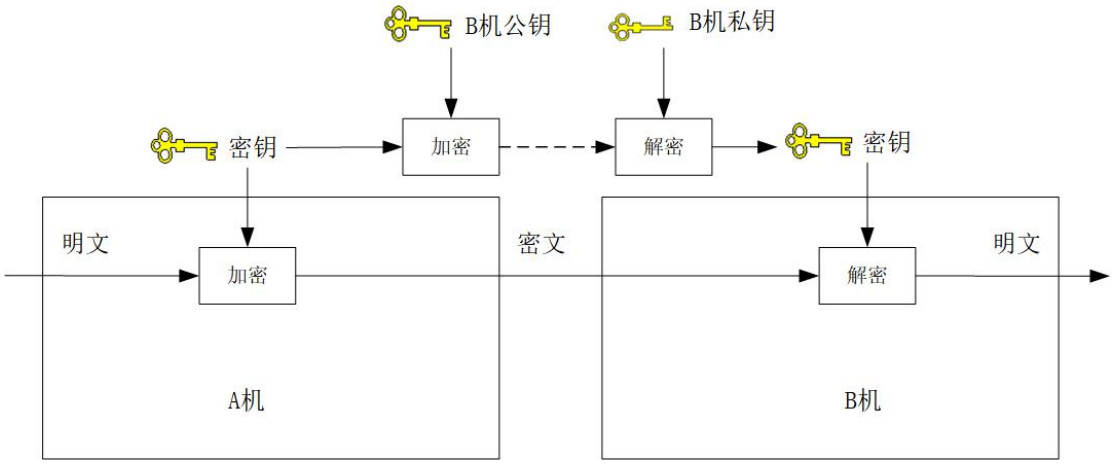

图 2-1 两台之间的加密通信示意图

2.2 程序的非功能需求

程序要求运行、反馈迅速，且上述两个步骤在程序中自动执行完，无手动参与；程序可以在同一台主机上完成，但数据必须经过网络传输。

RSA 和 AES 算法必须是源码编译得到，不能直接用编译过的库文件。

3 概要设计

本篇主要介绍相关技术、数据类型的定义、程序流程、主要类的介绍以及运行环境等。

3.1 相关技术介绍

两台之间的交互措施使用 socket 网络编程。

3.3.1 Sokect 介绍

Socket 是应用层与 TCP/IP 协议族通信的中间软件抽象层，它是一组接口。在设计模式中，Socket 其实就是一个门面模式，它把复杂的 TCP/IP 协议族隐藏在 Socket 接口后面，对用户来说，一组简单的接口就是全部，让 Socket 去组织数据，以符合指定的协议。其在因特网协议栈中的位置如图 1-2 所示。

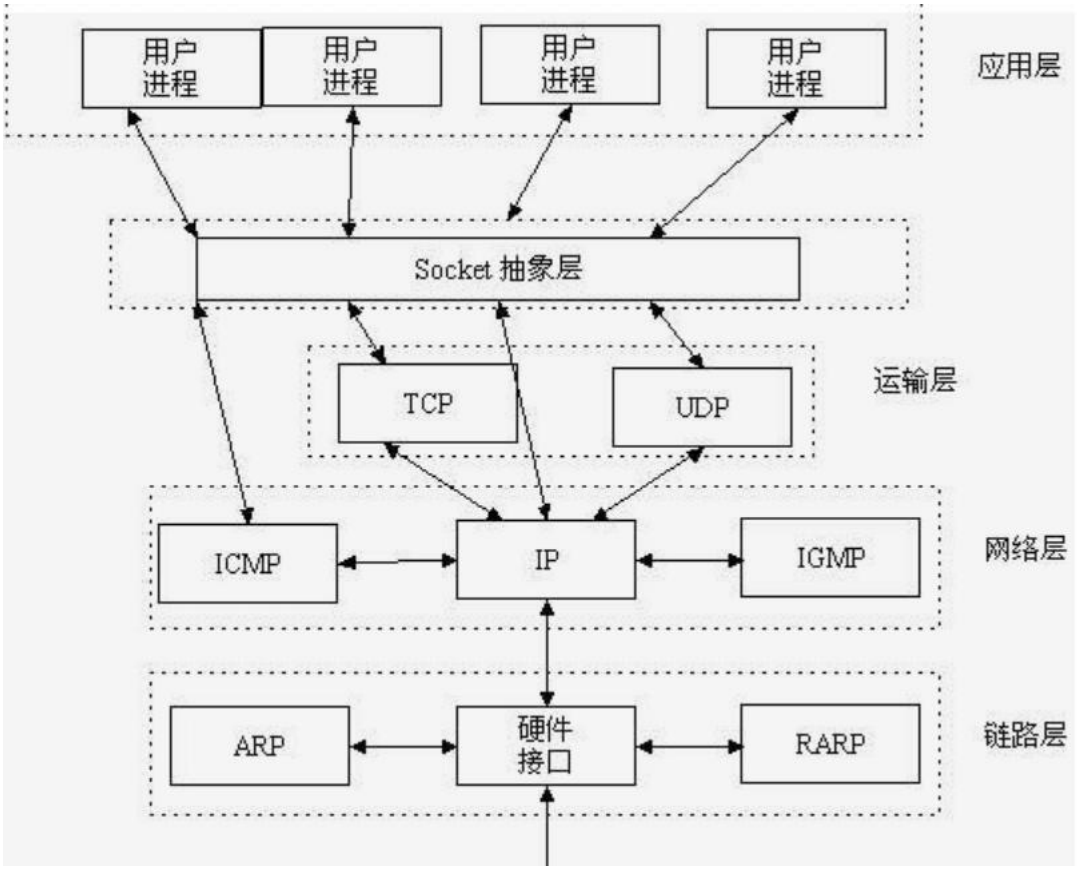

图 3-1 socket 在协议栈中的位置示意图

简单来说，通过 socket 技术，我们可以实现网络中进程之间的通信，通过这种协议+端口的模式，可以让两个进程相互传输数据。

3.3.2 RSA 加密技术

RSA 是一个典型的公钥密码（Public-key Cryptography），以当时在 MIT 的提出者 Rivest、Shamir 和 Adleman 三个人的名字命名，于 1978 年公开描述。RSA 既可以用于加密也可以用于签名（这其实是非对称密钥共有的属性），广泛用于电子商务安全系统中，当给定足够长度的秘钥时，RSA 被认为是安全的。

RSA 的加密系统的一般描述如下：存在用户 A、B。用户 A 随机选取两个素数 p 和 q，这两个素数应该具有相似的比特长度。计算出 n=pq，并记

$$\Phi(n)=(p-1)(q-1) \tag{1}$$

为欧拉函数。接着选取一个整数 e，该整数满足 1<e<φ(n)，且 e 与φ(n)互素，将 n 与 e 作为公钥指数发布。计算出满足(2)式的 d，通常这个 d 使用扩展欧几里得算法得到，将 d 作为私钥指数保密。

$$de \equiv 1 \bmod \Phi(n) \tag{2}$$

用户 B 使用得到的公钥信息加密明文 M 发送给用户 A。B 用户通常做以下工作：

①将消息 M 转换为一个整数 m(0<m<n)。

②计算 $c \equiv m^e (\bmod n)$，将 c 传输给用户 A。

用户 A 收到消息 c，计算 $m \equiv c^d (\bmod n)$，解开明文信息。

由于该过程中通常需要算出某个数的高阶幂 mod n，因此常常用快速幂和公式（3）来解决：

$$m^e \% n = (m \% n)^e \tag{3}$$

其他相关数论知识不再赘述。

3.3.3 AES 算法

随着 DES 算法安全性降低，人们研究出一种新的对称加密算法 AES。它基于替换置换网设计，没有使用 Feistel 网络结构，软硬件实现都很快。

关于 AES 的基本操作流程，首先，他允许使用分组和密钥以 128bit 为基础，以 8 字节倍数扩展长度，因此 192、256bit 分租（或密钥）就被组织成 4*6/4*8

矩阵。AES 中用 Nb、Nk、Nr 分别表示分组矩阵、密钥矩阵列数、轮数。其中轮数又和 Nb 和 Nk 有关。

AES 中解密和加密互为逆过程，密钥由主密钥经过密钥调度算法扩展。对于 128bit 的明文而言，迭代产生 10 组 128bit 轮密钥。AES 加密过程如下：

首先执行初始密钥混合：明文字节矩阵与初始主密钥矩阵直接相加。

接下来执行轮变换：以分组和密文长度都为 128 比特为例，共执行 10 轮变换，从第一轮到第九轮操作都相同，包括以下子步骤：

①字节替换：执行一个非线性替换操作，通过查表替换每个字节。

②行移位：状态矩阵的每一行以字节为单位循环移动若干字节。

③列混合：基于状态列的混合操作。

④轮密钥加：状态的没一个字节混合轮密钥。轮密钥也是由秘钥调度算法产生。

3.2 数据类型的定义

3.2.1 头文件

```
#include<winsock.h>
#include<iostream>
#include "AES.h"
#include "BigInteger.h"
#include "EncryptDecrypt.h"
#include "RSA.h"
#pragma comment(lib,"ws2_32.lib")
```

3.2.2 局部变量

```
//定义长度变量
int send_len = 0;
int recv_len = 0;
int len = 0;
```

```
//定义发送缓冲区和接受缓冲区
char send_buf[200];
char recv_buf[200];
//定义服务端套接字，接受请求套接字
SOCKET s_server;
SOCKET s_accept;
//服务端地址客户端地址
SOCKADDR_IN server_addr;
SOCKADDR_IN accept_addr;
initialization();
//填充服务端信息
server_addr.sin_family = AF_INET;
server_addr.sin_addr.S_un.S_addr = htonl(INADDR_ANY);
server_addr.sin_port = htons(9999);
//建立缓冲区
char buf2[64];
```

3.3 主要类介绍

3.3.1 类定义

AES 加密算法相关类：AES.h 和 AES.cpp

BigInteger 大整数类相关类：BigInteger.h 和 BigInteger.cpp

RSA 加密算法相关类：RSA.h 和 RSA.cpp

EncryptDecrypt 加密解密类：该类在 RSA 类的基础上又封装了一层，为了方便与另一个用户交互，或者是执行一些与 RSA 相关的操作，例如打印 RSA 参数等。

3.3.2 AES 类

该类为 AES 加密算法的相关类，其中包括对数组或字符串的加解密操作，

以及 unsigned char 数组和 char 数组相互转换的操作等。下面开始逐一介绍函数头。如表 2-1 所示。

表 3-1 AES 类中的函数介绍

序号	方法名	返回类型	参数类型	函数类型	方法功能说明
1	AES	无	unsigned char*	Public	AES 类的构造函数，参数为主密钥
2	~AES	无	无	Public	析构函数
3	Cipher	unsigned char*	unsigned char*	Public	加密，传入的数组大小必须是 16 字节
4	InvCipher	unsigned char*	unsigned char*	Public	解密，传入的数组也必须是 16 字节
5	Cipher	void*	void* input, int length	Public	可以传入数组，大小必须是 16 的整数倍，如果不是将会越界操作；如果不传 length 而默认为 0，那么将按照字符串处理，遇'\0'结束
6	InvCipher	void*	void* input, int length	Public	必须传入数组和大小，必须是 16 的整数倍
7	convertUnCharToStr	Void	char* str, unsigned char* UnChar, int ucLen	Public	Unsignedchar 数组类型转换成 char 数组类型
8	convertStrToUnChar	Void	char* str, unsigned char* UnChar	Public	Char 数组类型转换成 unsignedchar 数组类型
9	KeyExpansion	void	unsigned char* key, unsigned char	private	密钥扩展算法

			w[][4][4]		
10	SubBytes	void	unsigned char state[][4]	private	字节替换
11	MixColumns	void	unsigned char state[][4]	private	列混合
12	AddRoundKey	void	unsigned char state[][4]	private	轮密钥加
13	InvSubBytes	void	unsigned char state[][4]	private	解密时的字节替换
14	InvShiftRows	void	unsigned char state[][4]	private	解密时的行移位
15	InvMixColumns	void	unsigned char state[][4]	private	解密时的列混合

3.3.3 RSA 类

该类中包含有公钥和私钥的产生算法以及公钥和加/解密和私钥加/解密的算法。其函数头如表 2-2 所示。

表 3-2 RSA 类中的函数介绍

序号	方法名	返回类型	参数类型	函数类型	方法功能说明
1	RSA	无	无	Public	RSA 类的默认构造函数
2	~RSA	无	无	Public	析构函数
3	RSA	无	const unsigned	Public	RSA 重载的构造函数，参数是要生成的长度
3	init	void	unsigned char*	Public	初始化,产生公私钥对
4	encryptByPublic	BigInteger	const BigInteger &	Public	公钥加密
5	decryptByPrivate	BigInteger	const BigInteger &	Public	私钥解密
6	encryptByPrivate	BigInteger	const BigInteger &	Public	私钥加密
7	decryptByPublic	BigInteger	const BigInteger &	Public	公钥解密
8	operator <<	ostream &	ostream &, const RSA &	protected	输出相关数据

9	createOddNum	BigInteger	unsigned	private	生成一个大奇数,参数为其长度
10	isPrime	bool	const BigInteger &, const unsigned	private	判断是否为素数
11	createRandomSmaller	BigInteger	unsigned char state[][4]	private	随机创建一个更小的数
12	createPrime	BigInteger	unsigned char state[][4]	private	生成一个大素数,参数为其长度
13	createExponent	void	const BigInteger &	private	根据提供的欧拉数生成公钥、私钥指数

3.3.3 EncryptDecrypt 类

EncryptDecrypt 类在 RSA 类上又封装了一层，目的是为了便于打印或者重置 RSA 相关信息。除此之外，其中的一些函数也能便于网络传递信息的解析。其函数头如表 2-3 所示。

表 3-3　EncryptDecrypt 类中的函数介绍

序号	**方法名**	**返回类型**	**参数类型**	**函数类型**	**方法功能说明**
1	EncryptDecrypt	无	无	Public	EncryptDecrypt 类的默认构造函数
2	~EncryptDecrypt	无	无	Public	析构函数
3	Encrypt	char*	char*	Public	加密，调用 RSA 类的函数
3	decrypt	char*	std::string str	Public	解密，调用 RSA 类的函数
4	print	void	无	Public	打印 RSA 相关信息
5	reset	void	无	Public	重置 RSA 相关信息

6	setNandE	void	const std::string&	Public	分离 n 和 e
7	getEnvelop	void	char*, char*	Public	获得公钥加密的对称密钥和对称密钥加密的明文

3.4 程序流程

3.4.1 A 机流程

A 机（加密信息的服务机）程序流程图如图 3-2 所示。

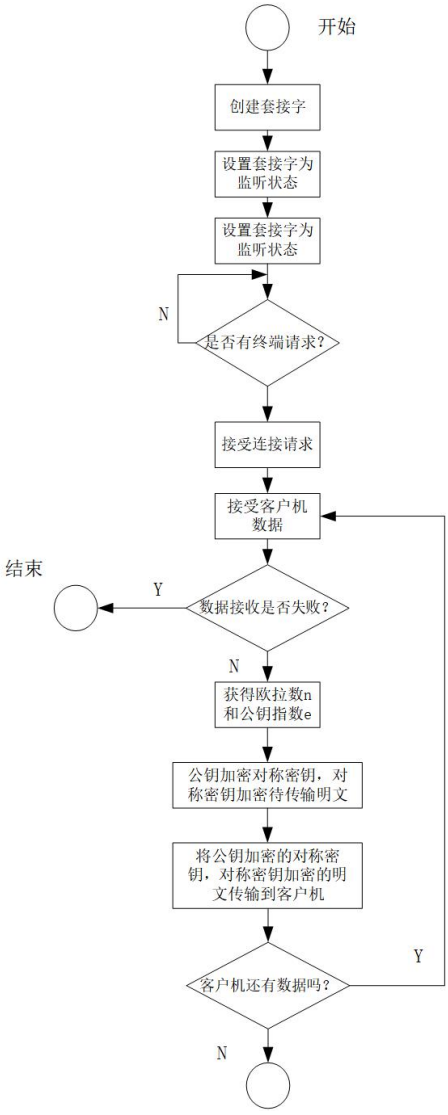

图 3-2 A 机总体流程图

3.4.2 B 机流程

B 机（接受加密信息的客户机）程序流程图如图 2-3 所示

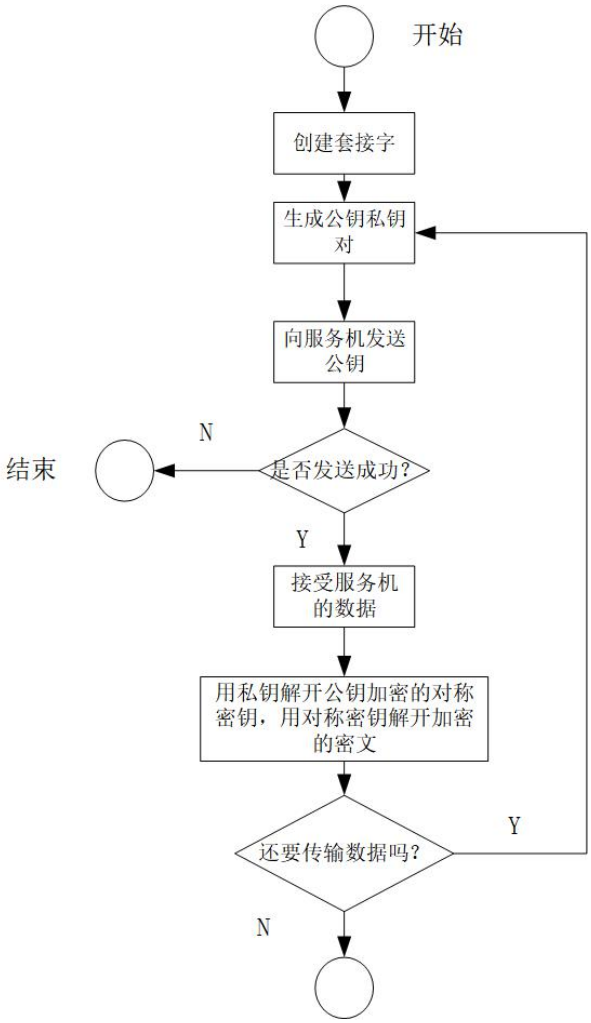

图 3-3 B 机总体流程图

3.5 运行环境介绍

硬件环境:PC 机内存 8G；硬盘 256G

软件环境：操作系统：windows10

编译环境：Visual Studio 2017 community

运行语言：C/C++

4 系统实现和结果展示

本工程中，AES 算法的密钥长度为 128 位，RSA 长度为 128 位。对称密钥为“12345678”，待传输的信息为“NPU”，其中对称密钥不满 128 的位置由 0 补齐。

进入服务机后，服务机进入阻塞状态，等待客户机的信息。如图 4-1 所示。

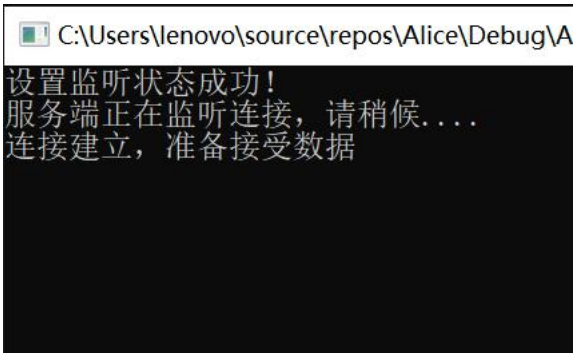

图 4-1 服务机处于阻塞状态

打开客户机，开始初始化公钥和私钥对。如图 4-2 所示。

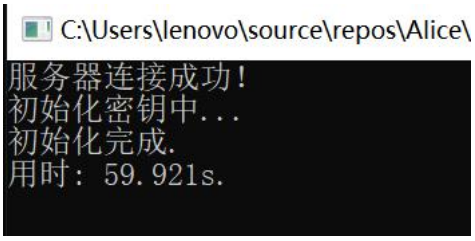

图 4-2 客户机初始化公钥私钥对

之后客户机将公钥信息发送服务机上，为了方便观察和证明，将对称密钥和加密后的对称密钥打在命令行中，如图 4-3 所示。之后将公钥机密的对称密钥和对称密钥加密的信息传送给客户机。

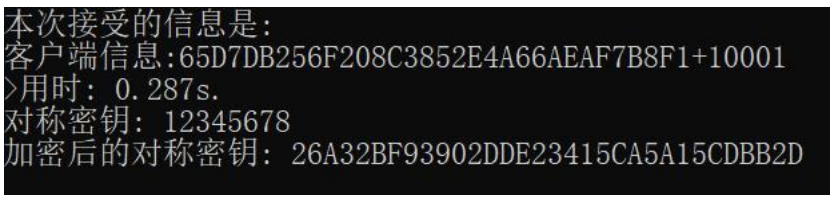

图 4-3 服务机接收到的公钥信息

客户机收到服务端信息，先用私钥解出加密的对称密钥，在用这把对称密钥解密加密的密文，最后解出明文如图 4-4 所示。

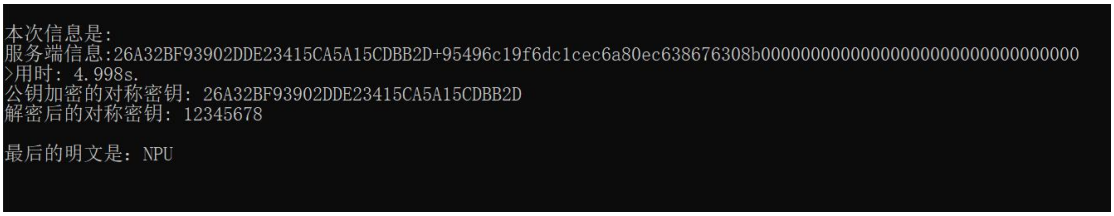
本次信息是:
服务端信息:26A32BF93902DDE23415CA5A15CDBB2D+95496c19f6dc1cec6a80ec638676308b00000000000000000000000000000000
>用时: 4.998s.
公钥加密的对称密钥: 26A32BF93902DDE23415CA5A15CDBB2D
解密后的对称密钥: 12345678
最后的明文是：NPU

图 4-4 客户机解出明文

5 项目的优缺点及展望

5.1 项目的优缺点

项目的优点：程序执行迅速，人机交互良好，且能重置公钥私钥对，实现公钥私钥的更新。实现了保密性、完整性、可认证性、可用性等方面的属性需求。

项目的缺点：该项目通过使用 AES 算法和 RSA 算法，能实现保密性、完整性、可认证性和可用性，但是却无法实现不可否认性。且无法自动更新。除此之外，事实上，应该将生成的公钥私钥对保存为二进制文件，在经过 base64 码加密以后再在网络上进行传输。

5.2 项目展望

①为了解决不可否认性，我们可以在 A 机也申请一对公钥和私钥，这对公钥和私钥用于数字签名。一般来说，由于需要传输的数据会很长，我们通常使用 Hash 函数来对明文生成摘要，并用 A 机的私钥对该摘要进行数字签名，之后，会将公钥加密对称密钥、对称密钥加密的信息，本机私钥签名的摘要一同发送给 B 机。B 机收到上述三份信息后，首先用自己的私钥解出对称密钥，再用对称密钥解出明文，并通过与 A 机相同的哈希函数得到摘要信息；之后用 A 机的公钥解开私钥签名的摘要，也得到了一份摘要信息，通过比较两份摘要信息，即可完成不可否认性认证。如图 5-1 所示。

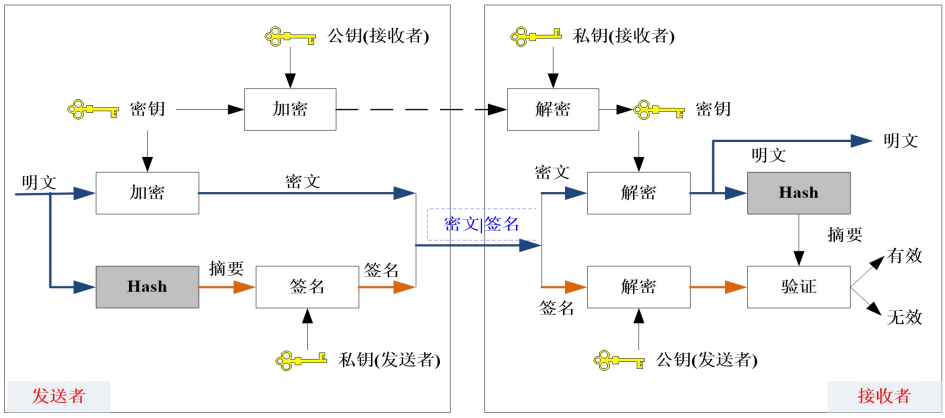

图 5-1 带签名的加密封装

②为了避免运算时间过长，可以设计一个缓冲队列，将下一次要更新的公钥私钥对提前算完，这样能有效的提高两台主机之间的交互速率，但是这样做具有一定的危险性。

③没有设计 UI 界面，可以利用微软的 MFC 库进行 UI 设计。

6 总结

本次工程通过使用 RSA 加密算法和 AES 加密算法，实现了两台主机之间的信息传输。其中 RSA 属于公钥密码体制，其优点为能很好的解决密钥的管理与分发问题，但是运算速度较慢；AES 算法属于对称密码体制，其优点是运算速度较快，但是难以解决密钥的分发问题。通过本工程的这种方法，结合两者的优点，解决了密钥分发难、加密速度慢等问题，实现了信息安全学科中保密性、完整性、可认证性、可用性等方面的属性需求。同时，在本文的展望中，也分析该工程并提出一些优缺点以及一些解决方法。例如可以通过数字签名解决不可否认性的问题，利用缓冲队列解决密钥更新时间长的问题等。

参考文献

[1]kunnyk.C++Socket 编程（一）概述

[EB/OL].https://blog.csdn.net/yangkunqiankun/article/details/75808401,2017-07-22.

[2]Elpsywk.AES 算法的 c++实现

[EB/OL].https://www.cnblogs.com/elpsycongroo/p/7830549.html,2017-11-14.

[3]Silenceneo_xw.RSA 加密算法 C++实现

[EB/OL].https://www.cnblogs.com/Silenceneo-xw/p/6718334.html,2017-04-19.

[4]张浩军,杨卫东,谭玉波.信息安全技术基础[M].中国水利水电出版社:北京市海淀区,2011