1. **实验目的**

设计一个保密通信的协议，具体要求为:利用RSA公钥密码算法，为双方分配一个AES 算法的会话密钥，然后利用AES加密算法和分配的会话密钥，加解密传送的信息。

1. **实验原理与协议设计**

本次实验设计了一个保密的通信协议，对称加密方案为AES -CBC模式加密，非对称加密方案为RSA加密，通信采用Windows Socket 基于TCP通信。RSA以及AES等加密算法使用OpenSSL库完成。

假定前提条件：

1.AB双方都已有各自的RSA公钥和私钥，并且互相知道对方的公钥。

这里实现方法是AB从本地文件中读取并解析已经制作好的公钥私钥文件。在每个程序的根目录下都有一个RSAkey文件夹，该文件夹里有3个文件，分别是自己的公钥文件、自己的私钥文件、对方的公钥文件。

2.AES加密所需的16字节密钥由A程序在内存中自动生成，并通过TCP通信和B完成AES密钥交换

SSL是Secure Sockets Layer（安全套接层协议）的缩写，可以在提供安全秘密性传输，它是在Socket的基础上结合现代密码学技术，对传输的内容进行加密，并对通信双方进行身份验证，将套接字层封装为一个更安全可靠的安全套接层。OpenSSL是一个符合目前工业标准的开源的和安全加密认证相关的软件库，开发人员可以在OpenSSL提供的软件接口的基础上，设计并实现自己项目所需的SSL。

AES是一种对称加密算法，其特点是加密后的数据长度和原数据长度一致，常用在对较大的明文进行加密，并且AES算法要求明文长度要向密钥长度对齐，即size(Plain)%size(key)==0, 因此进行加密前往往需要对明文进行填充padding,常用的填充算法为PKCS，这里本实验的AES-CBC加密算法的密钥长度为128位16字节，并采用PKCS7对明文进行填充至16字节的倍数。

RSA加密算法是非对称的，其一次加密过程对明文长度有限制，一般不能超过密钥的长度，即如果密钥为1024bit，则加密的数据不可以超过128字节，否则需要分块进行加密。RSA一般常用在对AES的密钥进行加密,进行通信双方的密钥交换过程。

协议设计：

1. 首先发送方A会向接收方B发起一个TCP连接。
2. 当连接建立成功后，A会生成一个128位的AES密钥key，并用B的公钥PubkeyB对key进行加密得到密文C\_key,并将此密文发送给B
3. 仅仅通过RSA加密是不够安全的，这里还需要进行身份认证，因为很有可能存在攻击者，通过滥用B的公钥，伪造A的身份向B发送混淆性的数据。因此这里加入数字签名机制：
4. 发送方A计算发送密文的SHA256值，得到256位的SHA值，然后对SHA使用A的私钥prikeyA进行加密，得到数字签名Sig,并把Sig一同发送给B。这样即便攻击者获取到了B的公钥，也会因为没有A的私钥而无法伪造身份。
5. 接收方B接收来自A的key密文C\_key和数字签名Sig。然后，B先用A的公钥对签名Sig解密得到SHA\_1,接着B再自己本地对密文C\_key计算一份SHA256值得到SHA\_2,然后B对SHA\_1和SHA\_2进行比对，如果二者一致，则签名验证通过，说明是A发送的数据。否则B认为此次通信不安全，立即断开连接。
6. 在通过B的验签后，B对C\_key使用自己的私钥进行解密，得到最终的AES key,至此，AES密钥交换过程完成

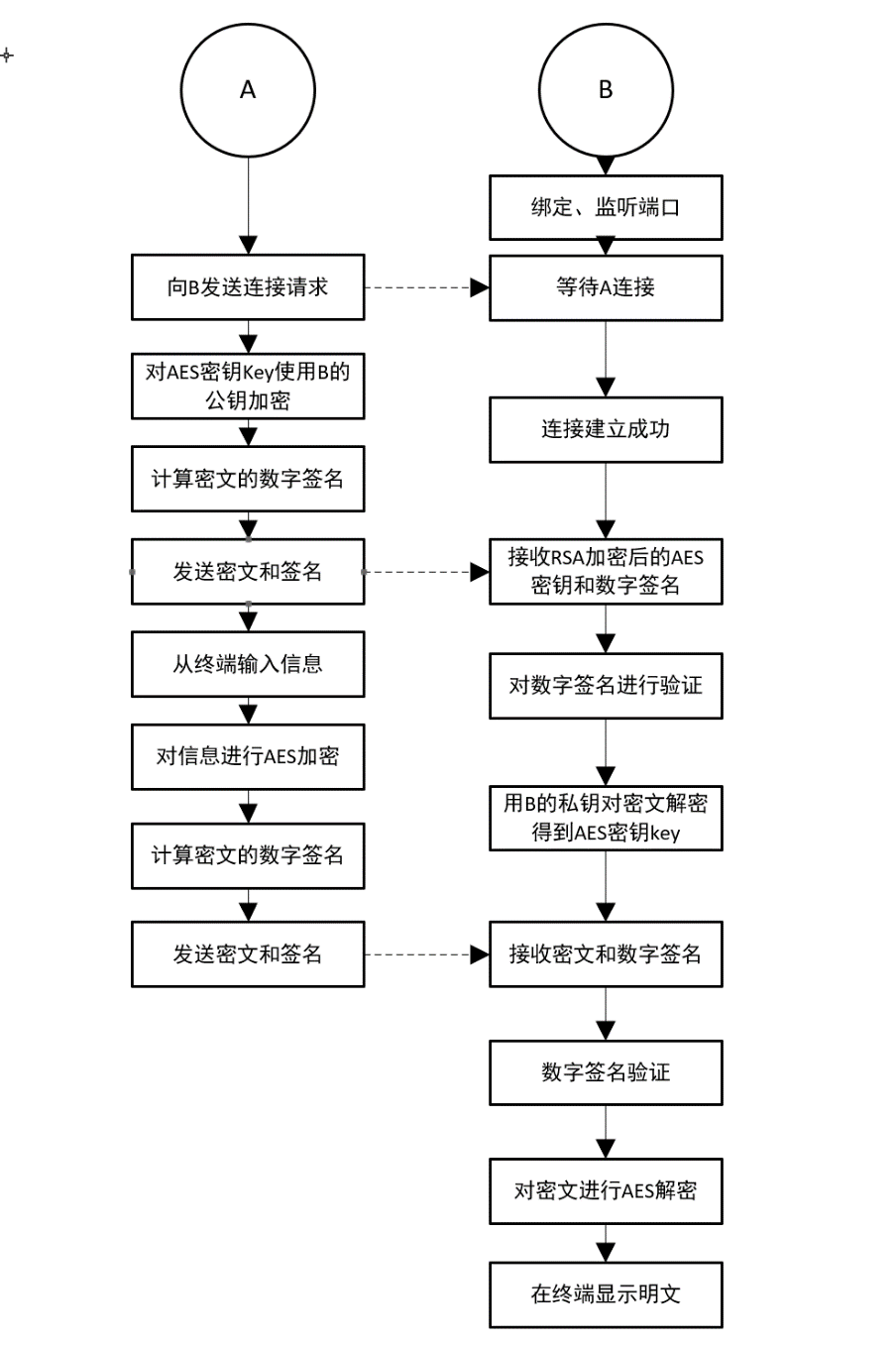
在此之后，发送方A可以持续向B发送AES加密后的密文C，并且考虑安全性，同样也要追加发送一份C的数字签名Sig。

B每次接收2份数据，一份为AES加密后的密文C，另一份为该密文对应的数字签名Sig,并且要对签名进行上述过程的验证，只有验证合法后，才继续解密密文，并输出显示

因此通过上述的基于RSA和AES的混合加密，配合数字签名验证机制，可以做到安全的可靠信息传输。

通信过程：通信采用Socket网络通信，在AB之间建立一条TCP连接，因为TCP是有连接的传输，通信过程更可靠一些。在本程序中为做演示，ip地址为本地环回127.0.0.1 通信端口号为54321。

程序的交互流程图如下：



1. **实验环境**

**操作系统：Windows10 64bit**

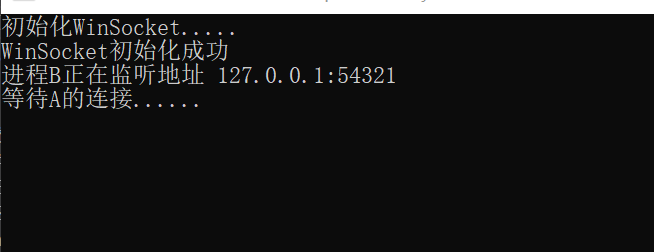
**编译环境：MSVC++、WinSock2、OpenSSL1.1**

**开发工具：Visual Studio**

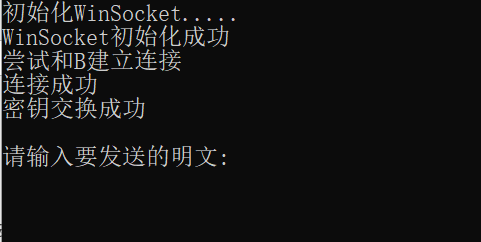
1. **运行测试**

由于本此实验的两个程序是基于TCP通信的，所以需要先让接收方B程序运行并绑定监听端口号（相当于一个Server），然后再启动发送方A程序向B发送数据（相当于一个Client）;并且本程序的通信所用的TCP端口号为54321，如果B无法正常绑定端口,则可能存在其他正在运行的程序占用了这个端口，解决方法为修改端口号，或者关闭正在占用该端口的程序。

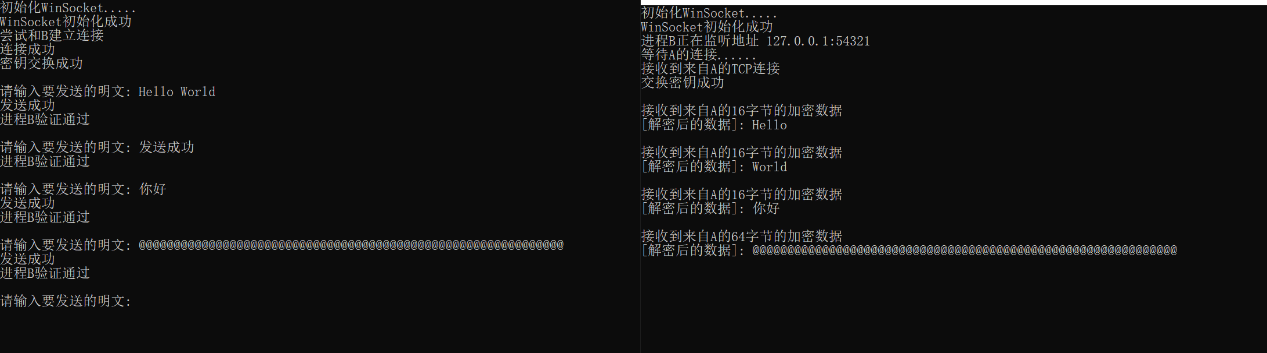
**1.先运行B,使其绑定并监听端口，等待A的连接**



2.再启动A,和B建立连接并交换AES密钥



3.接着，A可以不断从终端输入数据，A会把输入的数据按照协议中设计的那样进行加密传输。B会对密文进行解密并完成身份验证工作，最后会把明文显示在B的终端上：



1. **文件说明**

**VS工程目录：**

**该VS工程包含AB两个子工程**

**A: 发送方A的子工程目录，包含A工程的源代码、A所需的RSA密钥文件**

**B: 接收方B的子工程目录，包含B工程的源代码、B所需的RSA密钥文件**

**lib: OpenSSL的静态库版本**

**include: OpenSSL所需的包含目录（头文件）**

**如要编译该工程，请分别在两个子工程的 属性->VC++目录->包含目录中添加include文件夹，在 属性->VC++目录->库目录 中添加lib文件夹**

**可执行文件分开放在两个文件夹里面**

**A:**

**RSAkey:存放A的私钥、公钥，B的公钥**

**A.exe 发送方可执行文件**

**B:**

**RSAkey:存放B的私钥公钥、A的公钥**

**B.exe 接收方可执行文件**

**所有可执行文件均在Win10上用VisualStudio（MSVC）编译，编译模式为64bit Debug模式**

**关于PKCS1格式1024bit 的RSA公钥和私钥文件的生成，使用如下openssl命令：**

**1.生成2份私钥 Aprikey.pem Bprikey.pem**

**openssl genrsa -out Aprikey.pem 1024**

**openssl genrsa -out Bprikey.pem 1024**

**2.生成2份对应的公钥 Apubkey.pem Bpubkey.pem**

**openssl rsa -in Aprikey.pem -pubout -RSAPublicKey\_out -out Apubkey.pem**

**openssl rsa -in Bprikey.pem -pubout -RSAPublicKey\_out -out Bpubkey.pem**