模拟文件传输器

赵瑞霖 软件研究所 202228015059004

1.1 简介

本项目的主题是实验 1B，即应用层协议设计。本项目实现了一个基于 Qt 和 Windows Socket 套接字的文件传输协议。本项目加入了基于 RSA 的身份验证和 AES 的文件加密传输，能在一定程度上抵御常见的攻击。

1.2 开发环境

开发语言：C++
Qt 5.14.1
C++标准：C++11

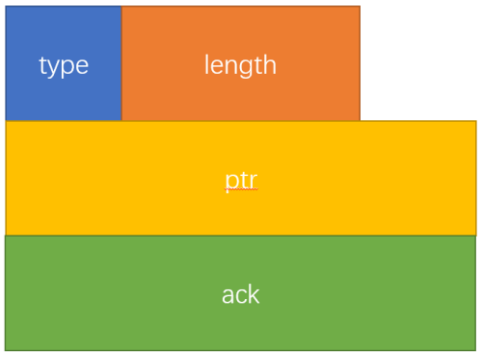
1.3 项目文件说明

FileTransfer.exe：程序主体。点击运行。
client-pri.key：客户端的私钥文件。
client-pub.key：客户端的公钥文件。
server-pri.key：服务端的私钥文件。
server-pub.key：服务端的公钥文件。
text.txt：用于传输实验的文本文件。
其余的文件皆为程序运行需要的动态链接库等。

1.4 协议说明

1.4.1 协议头部

项目所使用的应用层传输协议的报头结构如下：



type：1 字节，为报文的类型。
length：2 字节，为报文的长度（不包括报头）。虽然如此，为了防止缓存满造成崩溃，协议一次传输的最大字节量为 1024 字节。
ptr：4 字节，为报文的序号，具体功能类似 TCP 协议的 seq。
ack：4 字节，为发送端的确认号，具体功能类似 TCP 协议的 ack。
类型包括如下几种：
CONNECTION_REQUEST=0，请求连接报文，由客户端（主动传输文件的一方）

发出。

CONNECTION_REQUEST_AGREE=1，同意连接报文，由服务端（主动打开端口供客户端连接，被动接收文件的一方）发出。仅当接收到 CONNECTION_REQUEST 时会发出此报文。

FILENAME_TRANSFER=2，文件名传输报文，由客户端发出。传输客户端要发送的文件的名字。

DATA_TRANSFER=3，文件传输报文，由客户端发出。传输文件内容。

DATA_TRANSFER_END=4，文件传输结束报文，由客户端发出。代表客户端已经传输了所有的文件内容。

DATA_TRANSFER_END_AGREE=5，同意文件传输结束报文，由服务端发出。代表服务端已经成功接受了所有的文件内容，同意结束本次文件的传输。

IDENTIFICATION=6，身份信息传输报文，双方都可以发出。用于身份验证等内容的传输。

CONNECTION_BREAK=7，中止连接报文，双方都可以发出。发送这种报文是由于发送方出现了异常情况而主动中断连接。

KEY_SESSION=8，AES 密钥传输报文，由服务端发出。发送用于文件内容加密的 AES 密钥。

FILESIZE=9，文件大小传输报文，由客户端发出。向服务端传输应当接收的文件大小（单位：字节）。

DATA_TRANSFER_ACK=10，文件传输确认报文，由服务端发出。代表服务端收到了正确的文件内容，向客户端请求下一份数据。

1.4.2 安全机制

本协议采用以下的方法保证安全（假设 B 要给 A 传输文件）：

① A 和 B 知晓彼此的公钥。

② B 用 A 的公钥加密自己的身份信息以及标识符 NOUNCE1 发给 A。

③ A 获得了后用自己的私钥解密信息，将得到的 NOUNCE1 拼接上自己生成的 NOUNCE2 用 B 的公钥加密此信息发给 B。

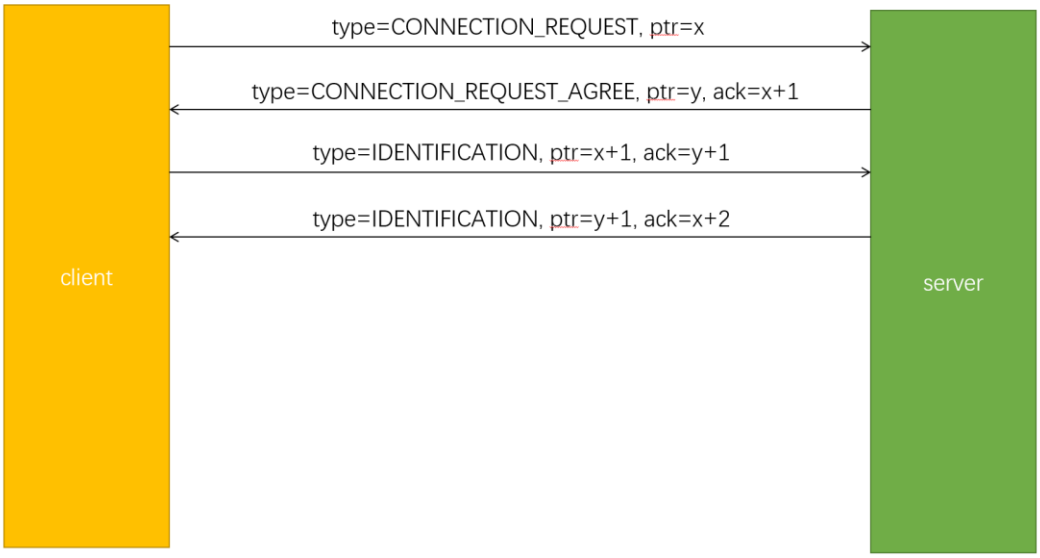
④ B 拿到后，用自己的私钥解密，得到 NOUNCE1，NOUNCE2，判断这个 NOUNCE1 和自己开始发出去的 NOUNCE1 是否一致，如果一致，则生成会话密钥 K，用自己的私钥加密（签名）。发送给 A。

⑤ A 用自己的私钥解开这个信息，再用 B 的公钥解密这个签名，得到会话密钥 K。用 K 作为密钥对文件内容加密并进行传输。

由上述可知，这是一个简化的条件。实际上，双方公钥的签发需要一个相对安全可信的中间方 CA 机构来执行，这个机构一般由政府等可信部门建立。为了简化条件，我们假设密钥是固定的，且双方已经知道了彼此的公钥，因此项目目录下存储着所有的公钥和私钥文件。此外，在实际情况下，用于文件内容加密和解密的密钥需要定期更新并广播，但为了简化条件，我们固定密钥的内容不变。

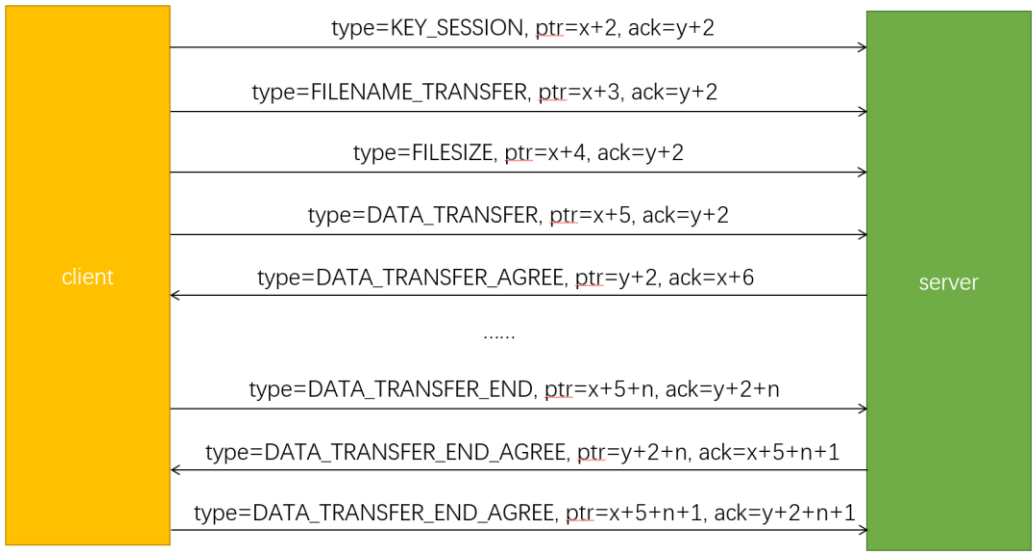
1.4.3 协议时序

本协议采用类似 TCP 建立及释放连接的方法定义时序。在建立连接阶段，双方建立 TCP 连接后，便开始应用层协议的连接，具体的时序如下图：



首先服务端需要打开端口让客户端进行 TCP 连接。在连接成功后，客户端发送一个请求连接的报文，服务端收到后回复一个同意连接的报文，接着客户端发送身份信息，服务端收到信息后加上自己的身份信息并返回，客户端检查返回的身份信息和自己的是否一致，如果一致，即连接成功。

在文件传输阶段，协议的时序如下：



为了安全性，客户端要给服务端传输用于文件加密解密的密钥文本，服务端以此生成对应的解密密钥（在多文件传输时，该过程只需一次即可，不需要每传输一次文件就要重新生成一次密钥）。然后客户端会依次传输文件名和文件大小供服务端记录并生成文件指针。

接着，客户端每发送一次数据包，服务端会进行接收，解密，写入文件并发送确认报文。客户端只有收到了确认报文，才会继续发送下一个数据包，在 TCP 协议的可靠传输下，这样就保证了服务端一定按序收到报文，获得文件的完整内容。重复此过程知道文件传输结束，给服务端发送一个文件传输结束的报文。服务端收到后，首先会判断已收到的数据量和文件大小是否一致，如果一致才会发

送同意文件传输结束的报文，文件传输结束。

如果客户端还想传输文件，则发送新的文件名，重复上述过程，不需要重发密钥。

若客户端想关闭连接，则客户端也返回一个 DATA_TRANSFER_END_AGREE 类型的报文，并主动关闭端口，服务端收到该报文后，也会关闭端口，双方就这样结束了连接。

1.5 操作方法

客户端和服务端共享一个界面，该界面如下：

其中，点击“打开端口（服务器专用）”，程序会变为服务端，打开端口；随后在客户端主机打开程序，首先输入目的 IP 地址，点击“连接（客户端专用）”，程序会变为客户端，与目标主机连接（默认打开 9000 端口）进行连接。接着就可以在“传输文件路径”中输入文件名，随后点击“传输（客户端专用）”传输文件了。下方的白色矩形为终端，程序内部的运行信息，包括是否连接成功，是否接受到目的报文，以及加密解密等步骤是否出现了错误，传输是否开始及完成，都会显示在此处。

若程序作为服务端使用，并打开了端口，在没有客户端主机连接时，程序会进入阻塞状态，此时无法活动。此外，在文件传输没有结束时，该程序也是不能执行其它操作的，只有等客户端主机传输结束，选择主动放弃连接时，服务端才可以进行其它操作。

1.6 其它

服务端最多支持 10 个并发的连接。

在程序的调试过程中，由于经常要进行本地的连接，且上传文件所在路径和文件传输的目的地路径是相同的，为了防止文件重名造成错误，程序规定：若服务端和客户端都在本地，则传输的文件会统一命名为“transferedData”，若两者不在同一主机上，则客户端传输的文件会在服务端按照相同的名称保存。

项目地址：https://github.com/0709ZRL/FileTransfer