## Http

HyperText Transfer Protocol，超文本传输协议，是互联网上使用最广泛的一种协议，所有WWW文件必须遵循的标准。HTTP协议传输的数据都是未加密的，也就是明文的，因此使用HTTP协议传输隐私信息非常不安全。

使用TCP端口为：80

## Https

Hyper Text Transfer Protocol over Secure Socket Layer，安全的超文本传输协议，网景公式设计了SSL(Secure Sockets Layer)协议用于对Http协议传输的数据进行加密，保证会话过程中的安全性。

使用TCP端口默认为443

## SSL协议加密方式

SSL协议即用到了对称加密也用到了非对称加密(公钥加密)，在建立传输链路时，SSL首先对对称加密的密钥使用公钥进行非对称加密，链路建立好之后，SSL对传输内容使用对称加密。

对称加密

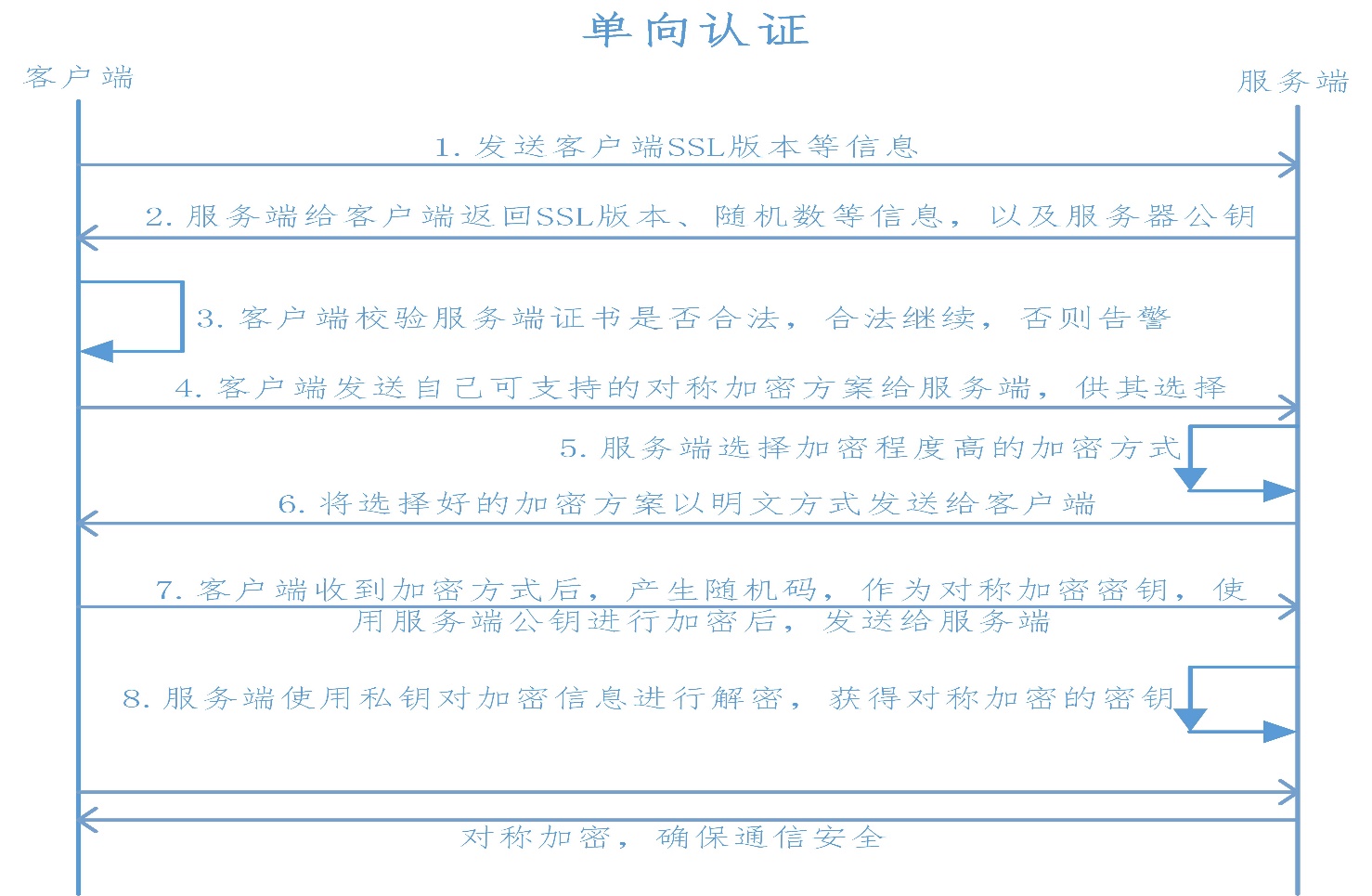
速度高，可加密内容较大，用来加密会话过程中的消息

公钥加密

加密速度较慢，但能提供更好的身份认证技术，用来加密对称加密的密钥

## 单向认证

Https在建立Socket连接之前，需要进行握手，具体过程如下：



客户端向服务端发送SSL协议版本号、加密算法种类、随机数等信息。

服务端给客户端返回SSL协议版本号、加密算法种类、随机数等信息，同时也返回服务器端的证书，即公钥证书

客户端使用服务端返回的信息验证服务器的合法性，包括：

证书是否过期

发型服务器证书的CA是否可靠

返回的公钥是否能正确解开返回证书中的数字签名

服务器证书上的域名是否和服务器的实际域名相匹配

验证通过后，将继续进行通信，否则，终止通信

客户端向服务端发送自己所能支持的对称加密方案，供服务器端进行选择

服务器端在客户端提供的加密方案中选择加密程度最高的加密方式。

服务器将选择好的加密方案通过明文方式返回给客户端

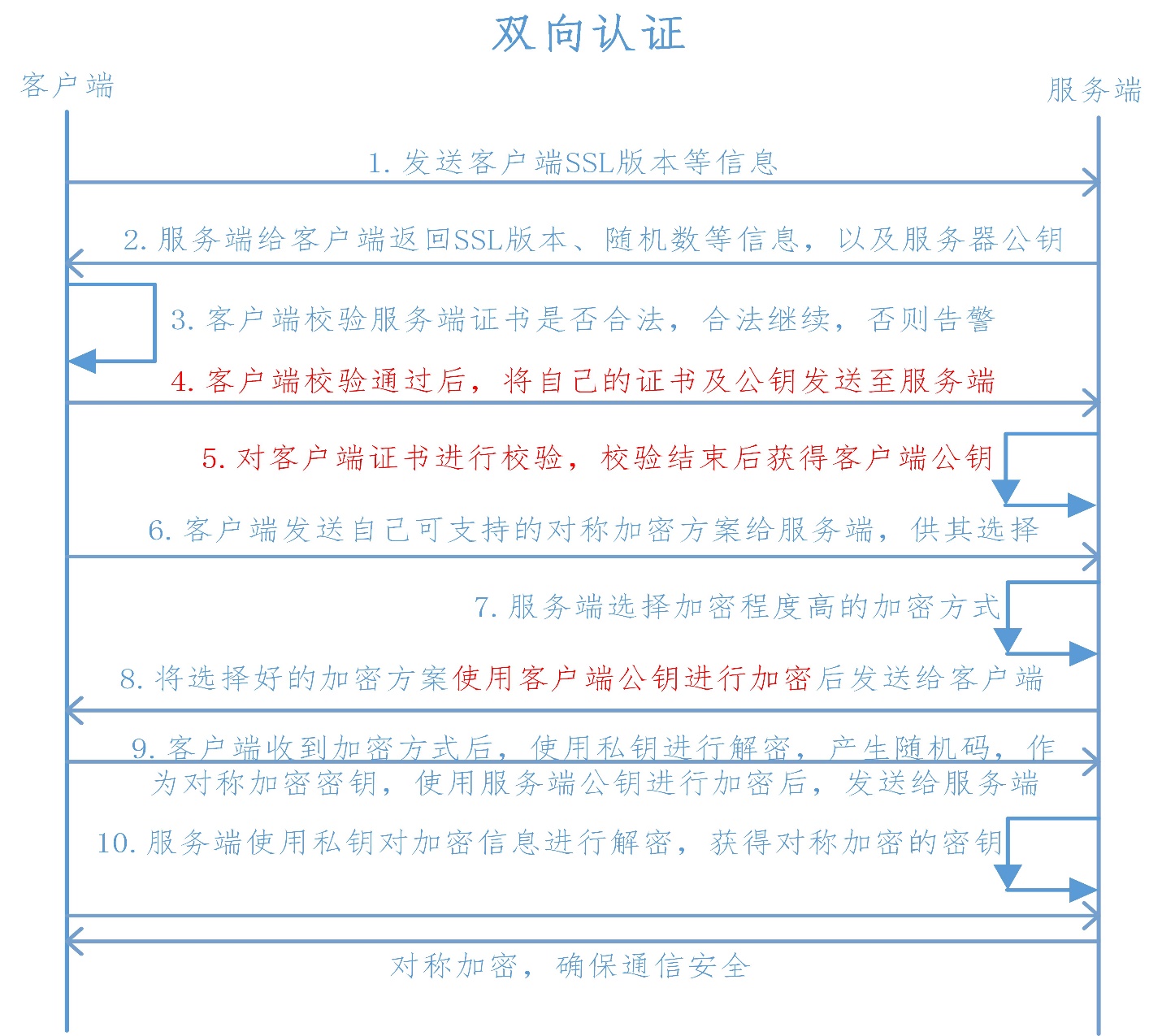
客户端接收到服务端返回的加密方式后，使用该加密方式生成产生随机码，用作通信过程中对称加密的密钥，使用服务端返回的公钥进行加密，将加密后的随机码发送至服务器

服务器收到客户端返回的加密信息后，使用自己的私钥进行解密，获取对称加密密钥。

在接下来的会话中，服务器和客户端将会使用该密码进行对称加密，保证通信过程中信息的安全。

五、双向认证

双向认证和单向认证原理基本差不多，只是除了客户端需要认证服务端以外，增加了服务端对客户端的认证，具体过程如下：



客户端向服务端发送SSL协议版本号、加密算法种类、随机数等信息。

服务端给客户端返回SSL协议版本号、加密算法种类、随机数等信息，同时也返回服务器端的证书，即公钥证书

客户端使用服务端返回的信息验证服务器的合法性，包括：

证书是否过期

发型服务器证书的CA是否可靠

返回的公钥是否能正确解开返回证书中的数字签名

服务器证书上的域名是否和服务器的实际域名相匹配

验证通过后，将继续进行通信，否则，终止通信

服务端要求客户端发送客户端的证书，客户端会将自己的证书发送至服务端

验证客户端的证书，通过验证后，会获得客户端的公钥

客户端向服务端发送自己所能支持的对称加密方案，供服务器端进行选择

服务器端在客户端提供的加密方案中选择加密程度最高的加密方式

将加密方案通过使用之前获取到的公钥进行加密，返回给客户端

客户端收到服务端返回的加密方案密文后，使用自己的私钥进行解密，获取具体加密方式，而后，产生该加密方式的随机码，用作加密过程中的密钥，使用之前从服务端证书中获取到的公钥进行加密后，发送给服务端

服务端收到客户端发送的消息后，使用自己的私钥进行解密，获取对称加密的密钥，在接下来的会话中，服务器和客户端将会使用该密码进行对称加密，保证通信过程中信息的安全。

## 配置https单向认证

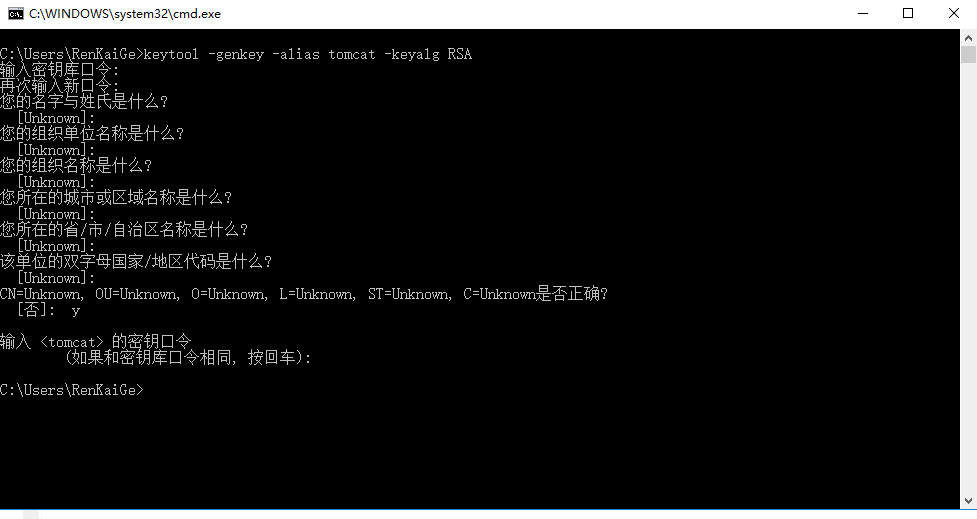
### 利用jdk自带命令执行

keytool -genkey -alias tomcat -keyalg RSA

会在用户的主目录中生成.keystore文件该文件是加密后的文件

window生成目录: C:\Users\RenKaiGe

linux生成目录:/root(linux中点开头文件是隐藏文件,需要使用ls –a查看文件)



### 将生成的.keystore文件复制到tomcat/conf文件夹下

在server.xml中将以下注释内容打开,并添加keystoreFile=”conf/.keystore”(keystore的相对路径或绝对路径), keystorePass=123456(创建keystore时输入的密钥库口令)

<Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"

maxThreads="150" SSLEnabled="true" scheme="https" secure="true"

clientAuth="false" sslProtocol="TLS" />

添加keystoreFile="conf/.keystore" keystorePass="123456"

<Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"

maxThreads="150" SSLEnabled="true" scheme="https" secure="true"

clientAuth="false" sslProtocol="TLS" keystoreFile="conf/.keystore" keystorePass="123456"/>

## 配置https双向认证