算法：RSA/DSA/SHA/MD5

　　非对称加密的算法有很多，比较著名的有RSA/DSA ，RSA可以用于加/解密，也可以用于签名验签，DSA只能用于签名。

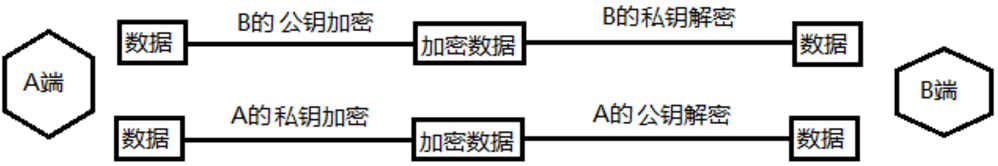
　　SHA是一种和MD5相同的算法，它不是用于加密解密或者签名的，它被称为摘要算法。就是通过一种算法，依据数据内容生成一种固定长度的摘要，这串摘要值与原数据存在对应关系，就是原数据会生成这个摘要，但是这个摘要是不能还原成原数据。这个算法起的作用就是验证数据有没有在传输过程中被修改了。

证书：CA/PEM/DER/X509/PKCS

　　一般的公钥不会用明文传输给别人的，正常情况下都会生成一个文件，这个文件就是公钥文件，然后这个文件可以交给其他人用于加密，但是传输过程中如果有人恶意破坏，将你的公钥换成了他的公钥，然后得到公钥的一方加密数据，他就可以用他自己的密钥解密看到数据了，为了解决这个问题，需要一个公证方来做这个事，任何人都可以找它来确认公钥是谁发的。这就是CA，CA确认公钥的原理也很简单，它将它自己的公钥发布给所有人，然后一个想要发布自己公钥的人可以将自己的公钥和一些身份信息发给CA，CA用自己的密钥进行加密，这里也可以称为签名。然后这个包含了你的公钥和你的信息的文件就可以称为证书文件了。这样一来所有得到一些公钥文件的人，通过CA的公钥解密了文件，如果正常解密那么机密后里面的信息一定是真的，因为加密方只可能是CA，其他人没它的密钥。这样你解开公钥文件，看看里面的信息就知道这个是不是那个你需要用来加密的公钥了。

　　实际应用中，一般人都不会找CA去签名，因为那是收钱的，所以可以自己做一个自签名的证书文件，就是自己生成一对密钥，然后再用自己生成的另外一对密钥对这对密钥进行签名

密钥文件的格式用OpenSSL生成的就只有PEM和DER两种格式，PEM的是将密钥用base64编码表示出来的，直接打开你能看到一串的英文字母，DER格式是二进制的密钥文件。X509是通用的证书文件格式定义。PKCS的一系列标准是指定的存放密钥的文件标准，PEM、DER、X509、PKCS这几种格式可以互相转化.



现在开始生成CA证书

创建私钥：genrsa -out ca-key.pem 1024

执行后生成ca-key.pem

创建证书请求：req -new -out ca-req.csr -key ca-key.pem

证书请求需要填写一些信息 如下：

// 国家名

Country Name (2 letter code) [AU]:CN

State or Province Name (full name) [Some-State]:BJ

Locality Name (eg, city) []:BJ

// 机构名

Organization Name (eg, company) [Internet Widgits Pty Ltd]:IE

Organizational Unit Name (eg, section) []:IE

Common Name (e.g. server FQDN or YOUR name) []:IE

// 邮箱

Email Address []:1373611035@qq.com

Please enter the following 'extra' attributes

to be sent with your certificate request

// 密码

A challenge password []:123456

// 公司名

An optional company name []:IECompany

执行后生成ca-req.csr

自签署证书：

x509 -req -in ca-req.csr -out ca-cert.pem -signkey ca-key.pem -days 365

执行完成之后会生成ca-cert.pem文件

将证书导出成浏览器支持的.p12格式：

pkcs12 -export -clcerts -in ca-cert.pem -inkey ca-key.pem -out ca.p12

生成server证书

创建私钥:

genrsa -out server-key.pem 1024

创建证书请求：

req -new -out server-req.csr -key server-key.pem -config openssl.cnf

自签署证书：

x509 -req -in server-req.csr -out server-cert.pem -signkey server-key.pem -CA ca-cert.pem -CAkey ca-key.pem -CAcreateserial -days 365

将证书导出成浏览器支持的.p12格式 ：

pkcs12 -export -clcerts -in server-cert.pem -inkey server-key.pem -out server.p12

生成Client证书（与服务器证书流程一样）