1. 加密算法
2. 对称加密

加密和解密的秘钥为同一个,如aes,des

1. 非对称加密

加密与解密秘钥不用，一般分公钥与私钥，公钥加密私钥解密，或者私钥加密公钥解密。不能用相同秘钥加密解密，如不能用私钥加密私钥解密或公钥加密公钥解密。

如RSA

1. 摘要算法： 如CRC32，MD5，SHA-1，SHA-256，SHA384，SHA-512
2. 对称加密效率高，非对称加密算法效率低，一般数据量大的内容用对称加密，然后对内容做摘要运算，对摘要做非对称算法加密，把对称算法的秘钥做非对称加密加密。接收着通过非对称算法解密得到对称算法的秘钥，再用此秘钥去解密数据。

二．数字证书的作用

1. 两个人Bob和Alice 通信， Bob可以使用对称秘钥key 对通信内容data加密，Alice

用相同秘钥进行解密。但这种非常不安全，例如James得知了秘钥key,他就也可以

解密信件数据data.

2. 以上用对称秘钥非常不安全，考虑用非对称秘钥，Bob生成了一对秘钥，把自己的

公钥BobPubKey给了Alice, Alice用Bob的公钥BobPubKey加密信件数据data,Bob可以用自己的私钥BobPrivateKey解密数据。同样Bob可以用Alice的公钥AlicePubKey加密数据给Alice,Alice用自己的私钥AlicePrivateKey解密数据.这种看起来是安全的，但是James还是有办法 ，他用自己的公钥JamesPubKey替换掉了Alice拥有的Bob的公钥，Alice并不知情，当她以为是用Bob的公钥加密数据时，其实是用了James的公钥JamesPubKey，这样当James截取数据后就可以用自己的私钥JamesPrivateKey 去解密获取内容然后伪装成Bob与Alice通信。

1. 为了避免上述的这种情况，数字证书被引入进来，有一个权威机构X可以确保公钥的拥有人。Bob提交自己的公钥BobPubKey给X,X把Bob的公钥连同Bob的个人信息组合打包，并用自己的私钥XPrivateKey给这个包做数字签名就成了Bob的数字证书BobCert。Alice 得到Bob的数字证书后，用X机构的公钥（X的数字证书（根证书）中获取）验证Bob的数字证书是否真的是Bob的，并且可以从Bob的数字证书中获取到Bob的公钥BobPubKey。如果验证成功，就可以用Bob的公钥通信了。如果这时James想假冒成Bob,替换了数字证书，但因为替换或的证书无法验证为Bob的，就会被Alice识别，欺骗失败，达到安全目的。根证书的数字签名是用根证书机构的私钥自己给自己签名，也就是自己颁发给自己。
2. 数字证书有两种编码，二进制编码DER,格式一般是.cer .crt ；Pem 格式编码，一般以.pem做后缀。仅包含公钥的数字证书。

.pfx和p12 采用二进制编码，包含个人数字证书和私钥，一般网银证书即采用此格式。

三．https

1. 采用SSL 协议，分为单向认证和双向认证
2. 单向认证是仅有客户端验证服务端的证书，当客户端访问服务器时，服务器会把进过颁发机构签名的服务器证书发送给客户端，客户端用根证书对服务器证书做验证，验证通过后，客户端和服务端协商出对称秘钥，后续双发通信及建立在此对称秘钥上。
3. 双向认证即服务器也要验证客户端的证书。
4. http basic 认证
5. 客户端首次请求服务器时，服务器会在返回的HEADER中返回错误吗401，并设置需要BASIC认证： response.setStatus(401);

response.setHeader("WWW-authenticate", "Basic Realm=\"dongxw\"");

1. 客户端浏览器在弹出的对话框中填入用户名和密码，浏览器自动使用如下引号内的格式“用户名：密码”做BASE64 变换数xxxxxx据后传给服务端。 Authorization: Basic xxxxxx
2. 服务端获取数据后做BASE64 解密，得到用户名和密码，进行验证。
3. 缺点明显，非常的不安全，很容易做BASE64破解。所以一下引入digest 认证
4. http digest 摘要认证：
5. 客户端首次请求服务器时，服务器会在返回的HEADER中返回错误吗401，并设置需要DIGEST认证： response.setStatus(401);

response.setHeader("WWW-authenticate", "Digest Realm=\"wudy\"") ；

1. 客户端浏览器在弹出的对话框中填入用户名和密码，浏览器根据用户名和密码以及其他的参数计算出response 返回给服务器。 Response 一般的计算方法如下: MD5(Hex2Ascii(MD5(username:realm:password)):nonce:Hex2Ascii(MD5(method:uri))) 其中nonce是服务器在HEADER中返回给客户端的现时随机数，method为请求方法“GET或POST”,url为域名地址。 例如 Digest username="abc", realm="wudy", nonce="", uri="/alpha/httpdigest-authz-server", response="a5ee16efea5532a4e8ffecfffc8961e5，密码为123， 则response= MD5(Hex2Ascii(MD5(abc:wudy:123))::Hex2Ascii(MD5(GET:/alpha/httpdigest-authz-server)))
2. 服务端获取用户名，在服务器上找到存储的密码和获取的其他相关参数通过相同的方法计算出response值是否与客户端传过来的相同，如果相同则认证通过。

六. http jwt(jason web token)认证