**一、加密和解密相关知识简介**

**1、信息安全标准**

NIST（National Institute of Standards and Technology）美国国家标准与技术研究院，制定了网络信息安全与保密的三个要素：

**保密性(confidentiality)：**信息不泄露给非授权用户、实体或过程，或供其利用的特性。（一般包括数据保密性、隐私性。）

**完整性(Integrity)：**数据未经授权不能进行改变的特性。即信息在存储或传输过程中保持不被修改、不被破坏和丢失的特性。（一般包括数据完整性、系统完整性。）

**可用性(Availability)：**可被授权实体访问并按需求使用的特性。即当需要时能否存取所需的信息。例如网络环境下拒绝服务、破坏网络和有关系统的正常运行等都属于对可用性的攻击；

（这三大要素被简称为：CIA）

尽管三要素能保证网络信息安全和保密，但从很多从事网络安全的研究人员的反馈发现，除了CIA外，还有另外两个标准也被经常提醒：

**真实性：**一个实体是真实的，是可被验证的。要确保数据发送方的确是它所声称的那个人。

**可追溯性：**一旦受到攻击，能追溯攻击发生的原处在什么地方。

**2、OSI组织定义的安全框架x.800**

**安全攻击：**

  被动攻击：窃听、（常见报文捕获、监听流量）

  主动攻击：伪装、重播、消息修改、拒绝服务

**安全机制：**

  加密、数字签名、访问控制、数据完整性、认证交换、流量填充、路由控制、公证

**安全服务：**

  认证 ：同等实体认证

  访问控制

  数据保密性：

    连接保密性

    无连接保密性

    选择域保密性

    流量保密性

**数据完整性** ：不允许插入、删除、修改、重播

**不可否认性**

**3、加密方式和算法**

（**1）对称加密：采用单钥密码系统的加密方法，同一个密钥可以同时用作信息的加密和解密。**

**对称加密的算法：**

    DES : 数据加密标准（56位密钥）

    3DES

    AES ：高级加密标准（128，192，256，384，512）

    Blowfish

    Twofish

    IDEA

    RC6

    CAST5

**对称加密的特性：**

    a）加密、解密使用同一口令；

    b）将明文分隔成固定大小的块，逐个进行加密

**对称加密的缺陷：**

   a）密钥过多；

   b）密钥传输；

  密钥交换、身份验正、数据完整性

**（2）公钥加密：由对应的一对唯一性密钥（即公开密钥和私有密钥）组成的加密方法。**

（公钥是从私钥中提取出来的。）

（公钥加密，只能私钥解密。私钥加密，也只能公钥解密。）

  密钥：public key, secret key  （p/s）

**常用加密算法：**

  RSA, DSA, EIGamal

  （DSA：只能用于身份验证）

**（3）单向加密：不可逆的加密**

**单向加密特性：**

**定长输出**: 无论原始数据是多大，结果大小都相同的

**雪崩效应:** 输入的微小改变，将会引起结果的巨大改变

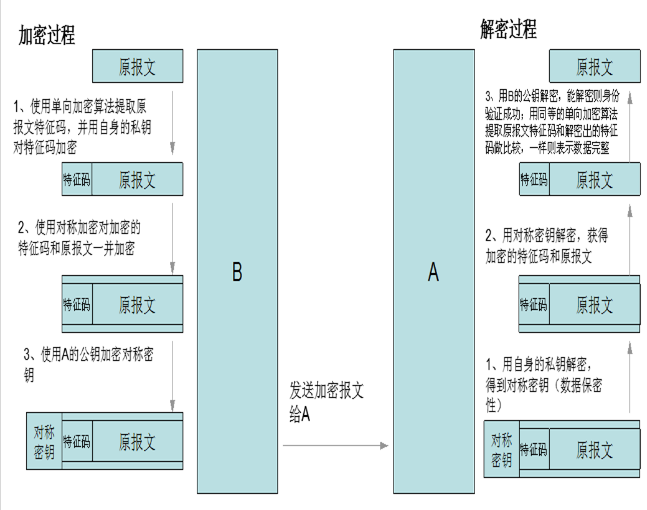
**单向加密算法**：MD5（128位）、SHA1、SHA256、SHA384、SHA512

**二、加密和解密的过程和原理**

 **首先问一个问题：假设B与A通信，B向A发送报文，怎么才能保证B的报文安全、可靠地被A接收到，并且保证报文数据的完整性？**

接下来围绕着这个问题来说明一下。

加密和解密的过程和原理如下图：



加密解密过程和原理详细说明：

**1、发送端B :**

（1）为保证安全，要对报文加密。加密方法有三类：对称加密、公钥加密和单向加密。对称加密不安全，单向加密是不可逆的，因而使用公钥加密。

  问题：公钥加密安全（一般为2048位），但是加密过程太慢了，不适用当前网络需求，该怎么办？

（2）为了解决上述问题，B可以用单向加密提取出报文的特征码（特征码能保证报文的数据完整性），再使用自身的私钥对特征码进行公钥加密（特征码数据小，对其进行公钥加密速度快），并把加密后的特征码附加到报文后。（使用私钥加密是为了验证身份）

问题：这种方式能实现数据完整性和身份验证的检验，但是却缺失了报文的数据保密性，又该怎么办？

（3）为了解决上述问题，B在把加密的特征码附加到报文后，把特征码和报文当做一个数据（假设为data），使用对称加密算法对该数据(data)加密得出一个密码，再把密码附加到该数据(data)后。为了使得在传输过程中密码不被其他人获取或篡改，使用A的公钥对密码进行加密（只有A的私钥能对其解密），把加密的密码附加到数据data后，再这些数据一并发送给A。

**2、接收端A：**

（1）A接收到B传来的报文，利用自身的私钥对其解密，获得密码。因为只有A的私钥能对B传来的报文（使用A的公钥加密密码）解密，所以能防止其他人对该传输的报文进行解密而获得其中的信息，保证了数据的保密性。

（2）A利用获得的密码解密其中对称加密的数据，获得经过加密的特征码和原报文。

（3）A使用B的公钥对该特征码解密，能解密则说明该报文是B发送过来的，实现了身份验证。（假设解密后的特征码是fcode）

（4）A使用同等单向加密算法对接收到的原报文提取其特征码。使用该特征码和解密后获得的特征码(fcode)做比较，如果一样，则说明原报文的数据完整。

问题：以上这种方式能保证数据完整性、身份验证和数据的保密性，在加密和解密的过程中都要用到对方的公钥，如何在传输过程中安全可靠地获得对方的公钥就成了关键的一环，那该如何做呢？

答：安全可靠地获取对方的公钥靠CA(Certificate Authority )证书授权中心来实现。

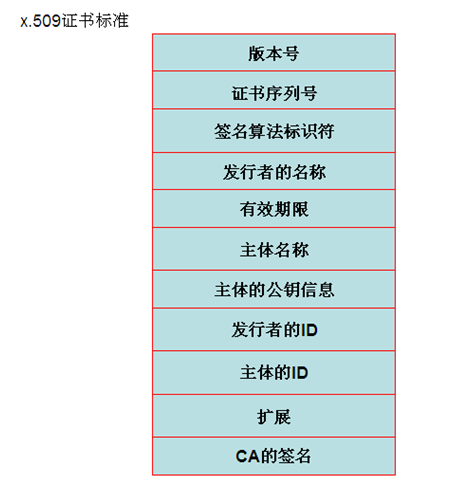
因而接下来，我们来说说CA。

**三、CA（证书授权中心）**

**1、CA证书标准：x.509**

x.509: 定义了证书结构和认证协议标准；（基于公钥和数字签名）

用于：IP安全、TLS/SSL（传输层安全）和S/MIME（安全电子邮件通信）



x.509证书标准详细说明：

（1）版本号（默认为1，如果有多个扩展，可能为3）

（2）证书序列号（是一个整数，在CA中唯一标识，表明发行了多少个证书）

（3）算法参数 （标志用了那种算法）

（4）发行者的名称（CA自己的名字）

（5）有效期限

（6）主体名称（证书拥有者名称）(很关键！！！)（个人用户使用的是个人用户名，主机使用的必须是主机名而不是ip地址）

（7）公钥（最重要）（公钥由证书拥有者提供）

（8）发行者的ID（CA的唯一编号）

（9）主体的ID（CA生成的证书拥有者唯一编号）

（10）扩展

（11）CA的签名（用于验证CA的来源合法性）

CA是相对于发送方B和接收方A的第三方，是具有公信力的机构。

**2、验证数字证书的过程**

B在发送之前获得A的数字证书或A在接收之前获得B的数字证书，都会去验证该数字证书的真伪。

以B在发送之前获得A的数字证书为例，说明**验证数字证书的过程**：

（1）要用对应给A发数字证书的那个CA的公钥去解密CA的签名，如果能解密，则说明A的数字证书确实是那个信任的CA所颁发的证书。

（2）解密出一段特征码，B再使用同样的单向加密算法提取A的数字证书的特征码，比较这两个特征码是否一样，如果一样，则表示获得的A的数字证书是完整的。

（3）此后，还要去验证该数字证书中的持有者是不是A，如果验证通过，才可以确定该数字证书确实是A的数字证书。

（4）确认该数字证书的拥有者是A后，还要去查看该数字证书是否在有效期限内和是否在CA的数字证书吊销列表中。

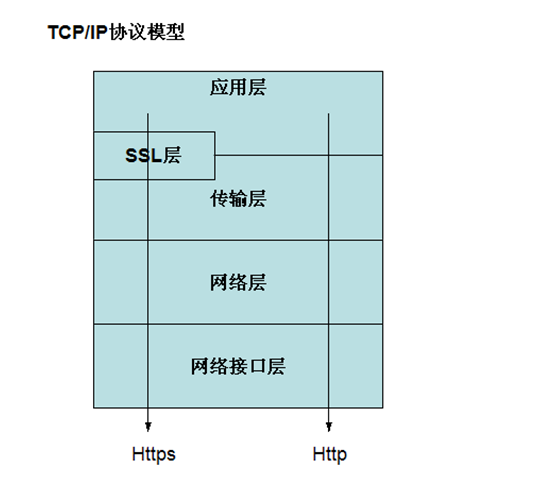
**四、SSL层**

**1、SSL层**

SSL(Secure Sockets Layer 安全套接层),及其继任者传输层安全（Transport Layer Security，TLS）是为网络通信提供安全及数据完整性的一种安全协议。

版本：sslv1, sslv2, sslv3

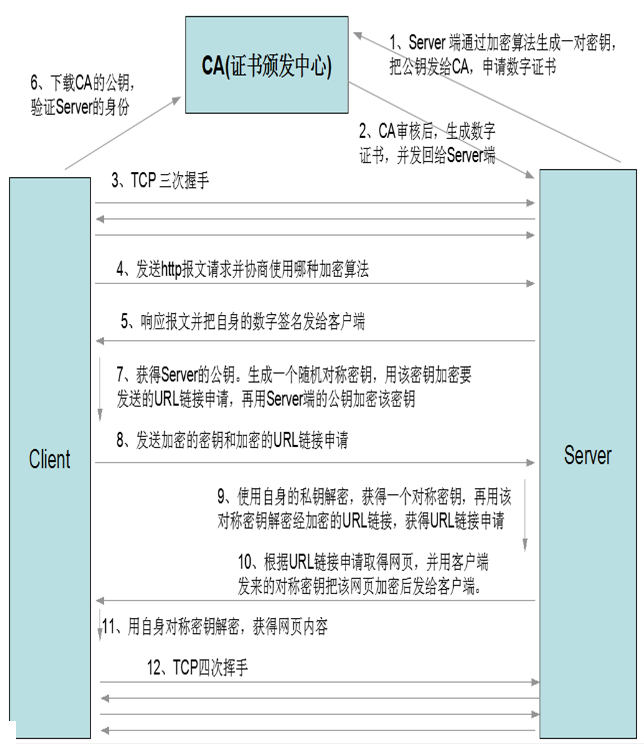
（ssl是介于网络层和传输层之间的半层，一般被制作成公共共享库，要想使用ssl就要调用ssl共享库。）



**2、https通信过程**

HTTPS（全称：Hyper Text Transfer Protocol over Secure Socket Layer），是以安全为目标的HTTP通道，简单讲是HTTP的安全版。即HTTP下加入SSL层，HTTPS的安全基础是SSL。

**以https为例，进一步说明如何依靠CA来可靠的获得通信对方的公钥**



https的主要实现过程说明：

（1）在通信之前，服务器端通过加密算法生成一对密钥，并把其公钥发给CA申请数字证书，CA审核后，结合服务端发来的相关信息生成数字证书，并把该数字证书发回给服务器端。

（2）客户端和服务器端经tcp三次握手，建立初步连接。

（3）客户端发送http报文请求并协商使用哪种加密算法。

（4）服务端响应报文并把自身的数字签名发给服务端。

（5）客服端下载CA的公钥，验证其数字证书的拥有者是否是服务器端（这个过程可以得到服务器端的公钥）。（一般是客户端验证服务端的身份，服务端不用验证客户端的身份。）

（6）如果验证通过，客户端生成一个随机对称密钥，用该密钥加密要发送的URL链接申请，再用服务器端的公钥加密该密钥，把加密的密钥和加密的URL链接一起发送到服务器。

（7）服务器端使用自身的私钥解密，获得一个对称密钥，再用该对称密钥解密经加密的URL链接，获得URL链接申请。

（8）服务器端根据获得的URL链接取得该链接的网页，并用客户端发来的对称密钥把该网页加密后发给客户端。

（9）客户端收到加密的网页，用自身的对称密钥解密，就能获得网页的内容了。

（10）TCP四次挥手，通信结束。

五、openssl自建CA过程详解

**OpenSSL**是套开放源代码的软件库包，实现了SSL与TLS协议。其主要库是以C语言所写成，实现了基本的加密功能。

OpenSSL可以运行在绝大多数类Unix操作系统上（包括Solaris，Linux，Mac OS X与各种版本的开放源代码BSD操作系统），OpenVMS与Microsoft Windows。它也提供了一个移植版本，可以在IBM i（OS/400）上运作。

此软件是以Eric Young以及Tim Hudson两人所写的SSLeay为基础所发展的，SSLeay随着两人前往RSA公司任职而停止开发。

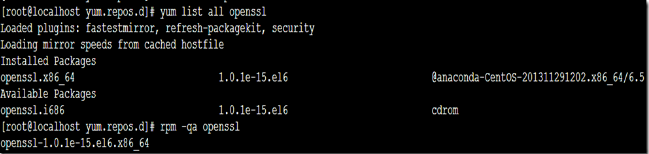
虽然此软件是开放源代码的，但其授权书条款与GPL有冲突之处，故GPL软件使用OpenSSL时（如Wget）必须对OpenSSL给予例外。

openssl建立私有CA的详细过程如下：

**openssl创建私有CA的过程：**

前提：

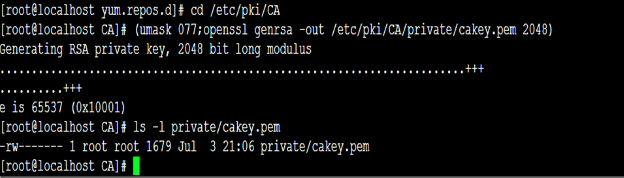
安装openssl ：# yum install openssl



1、建立CA服务器：

（1）生成密钥

# (umask 077; openssl genrsa -out /etc/pki/CA/private/cakey.pem 2048)



**命令解释**：

（）表示将会在当前shell中新建一个子shell，将（）中的命令放到该子shell中执行，执行完毕后关闭子shell并回到当前shell。

由于要对生成的cakey.pem文件设置合适权限，可使用umask修改文件的默认权限设置。为了不影响当前shell的默认权限设置，使用（）将这些命令放到子shell中执行就行了！

genrsa  : 指定使用rsa算法生成私钥

-out ：指定生成的私钥的存放位置（注意：该存放位置是在配置文件中默认定义了的，路径和文件名不能随意修改！！！）

2048 ：指定生成一个2048位的私钥

（2）自签证书

# openssl req -new -x509 -key /etc/pki/CA/private/cakey.pem -out /etc/pki/CA/cacert.pem -days 3655



**命令解释：**

req: 生成证书签署请求

-news: 新请求

-key /path/to/keyfile: 指定私钥文件（req命令能根据私钥自动抽取出公钥）

-out /path/to/somefile: （注意：路径和文件名不用随意修改！）

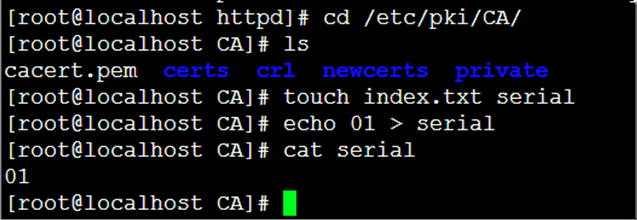
-x509: 专门用于生成自签署证书

-days n: 有效天数（一般和-x509一起使用才有意义。）

（3）初始化工作环境（只有第一次创建CA时，才需要初始化工作环境）

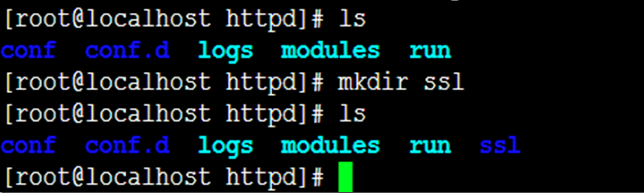
# touch /etc/pki/CA/{index.txt,serial}

# echo 01 > /etc/pki/CA/serial   （指定序列号从那个数字开始）



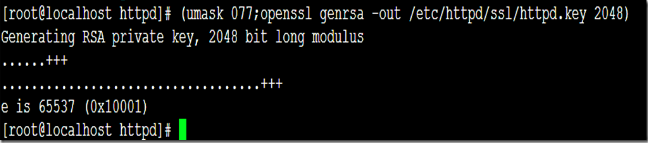
2、节点申请证书：

(1) 节点生成请求



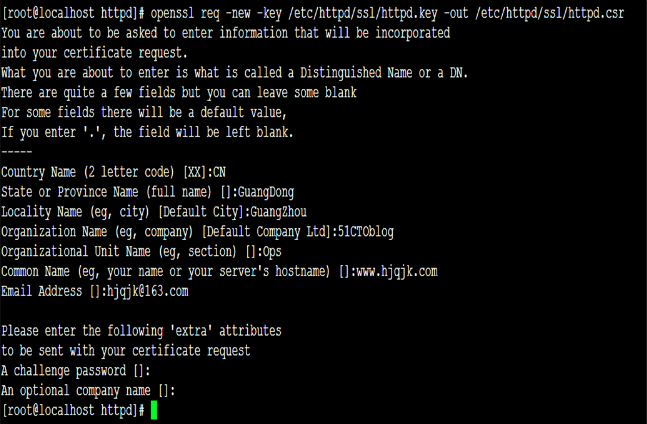
a、生成密钥对儿

# (umask 077; openssl genrsa -out /etc/httpd/ssl/httpd.key 2048)



b、生成证书签署请求

# openssl req -new -key /etc/httpd/ssl/httpd.key -out /etc/httpd/ssl/httpd.csr



-csr ：证书签署请求，一般都是这样的后缀

c、把签署请求文件发送给CA服务

(2) CA签署证书

a、验正证书中的信息；

b、签署证书

# openssl ca -in /path/to/somefile.csr -out /path/to/somefile.crt -days N

[root@localhost CA]# openssl ca -in /etc/httpd/ssl/httpd.csr -out /etc/httpd/ssl/httpd.crt -days 1000       
Using configuration from /etc/pki/tls/openssl.cnf        
Check that the request matches the signature        
Signature ok        
Certificate Details:        
        Serial Number: 1 (0x1)        
        Validity        
            Not Before: Jul  3 14:07:23 2014 GMT        
            Not After : Mar 29 14:07:23 2017 GMT        
        Subject:        
            countryName               = CN      #国家名  
            stateOrProvinceName       = GuangDong   #省份名     
            organizationName          = 51CTOblog   #公司名  
            organizationalUnitName    = Ops         #部门名  
            commonName                = www.hjqjk.com   #主机名     
            emailAddress              = hjqjk@163.com   #邮箱     
        X509v3 extensions:        
            X509v3 Basic Constraints:         
                CA:FALSE        
            Netscape Comment:         
                OpenSSL Generated Certificate        
            X509v3 Subject Key Identifier:         
                F9:DB:00:04:8A:D7:17:C8:21:B7:2D:15:F2:E9:89:66:BB:6D:D5:F9        
            X509v3 Authority Key Identifier:         
                keyid:98:56:B3:30:B0:9D:75:A1:69:AD:BF:2F:E4:0D:FE:3F:17:87:B0:A8  
Certificate is to be certified until Mar 29 14:07:23 2017 GMT (1000 days)       
Sign the certificate? [y/n]:y  #询问是否签署证书  
1 out of 1 certificate requests certified, commit? [y/n]y       
Write out database with 1 new entries        
Data Base Updated        
[root@localhost CA]# ls /etc/httpd/ssl      #证书已签署，自建CA到这里就成功了  
httpd.crt  httpd.csr  httpd.key

c、发送给请求者；

之后，只要把签署的证书发回给申请者就行了！