Openssl和OpenSSH

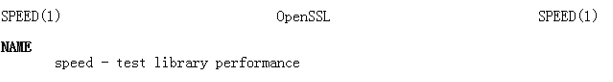
Openssl有三部分组成：

Libcrypto: 通用加密解密库 ---可以被软件开发调用

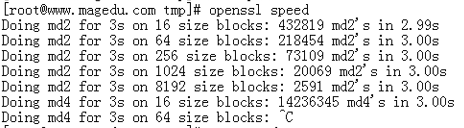
Libssl:实现TLS或者SSL 等功能的库

Openssl:多功能的命令行工具

标准命令：enc, dgst, genrsa, rsautl, req, ca, rsa, passwd, speed

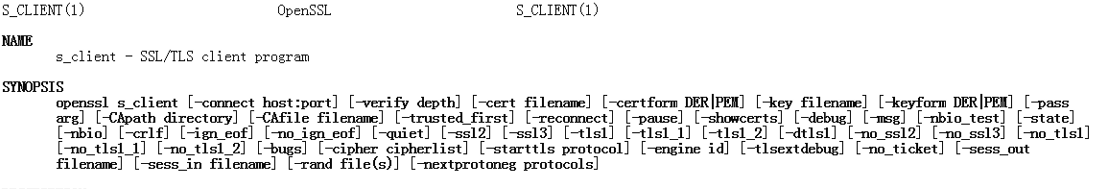


测试加密组件或者工具的性能 ---- 会在当前主机测试加密算法的执行速度



Version是查询openssl的版本的





这个是ssl/tls的客户端测试工具

学完Http协议之后 --- **基于文本的协议 ---- 这种基于文本的协议都可以使用telnet来做远程连接的访问测试的** ---万一 远程的协议不是http 而是https协议了 ---- 这个时候 就不能使用telnet了 要使用s\_client这个工具 将来会用到 可以用来执行



-connect表示连接的主机 -cert表示连接的时候 使用的证书

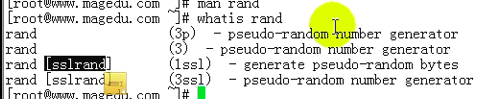
这个就是密钥

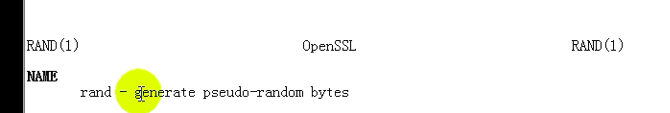
 ---这个是使用CA的证书文件

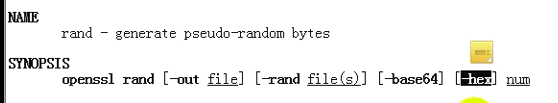
并且还可以在链接的时候 使用哪一个SSL协议

这个有个基本印象就可以了







这个是生成伪随机数 用来生成随机数字的  


可以使用-base64输出 或者-hex编码输出

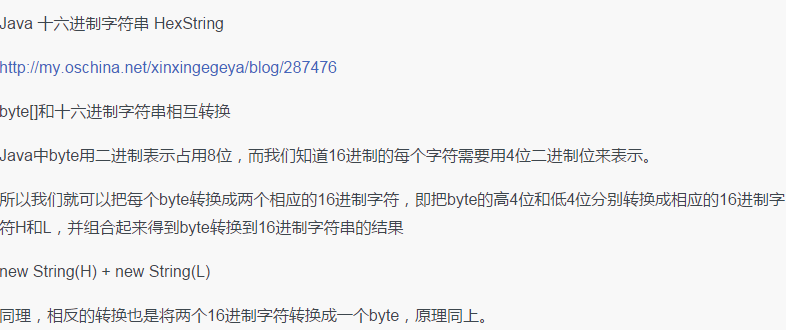


乱码

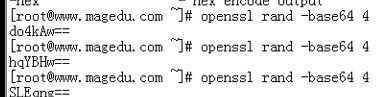
-hex是十六进制编码 ----- 十六进制编码 每4bit二进制是一个十六进制字符 ---- 这样一个字节是8bits ----应该显示为2个十六进制的字符

这样给出4 长度为4的二进制 --- 二进制是0100这四个字符（字符串的形式）---- 注意 是显示出来的字符 是4 一个显示的字符是一个字节 ---- 这样 每一个字节是2个十六进制字符----- 十六进制编码 ----- 显示8个











长短格式独特

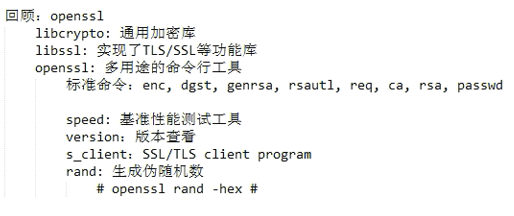


加入现在要获取数字

使用openssl passwd -1 –salt

这个盐 可以使用十六进制的





Openssl rand –hex ${number\_bytes} ---- 显示为多少个字节的伪随机数

上面试openssl剩余可能用到的命令

------ openssl可以进行证书的颁发 实现私有CA 格式是x509

X509证书格式：

公钥及其有效期限

持有者合法信息（用户或者主机）

证书的使用方式

CA的信息

CA的数字签名

---- 总结 ------

对称加密 来解决数据的私密性（也就是数据加密）--- 但是无法解决密钥交换 和 身份认证

公钥加密 用来实现密钥交换（对方的公钥） 数据加密（但是公钥加密的速度慢 通常不用 ---- 对方的公钥）身份认证（自己的私钥 ---- 加密的不是什么密钥 否则这样都会使用公钥解密出来 这样私钥加密的密钥就暴露了 所以 可以解密出来的结果 一定是不可逆的结果 通畅是数据的摘要信息）

一般而言 只要用公钥加密 是对方的公钥 私钥加密 都是自己的私钥

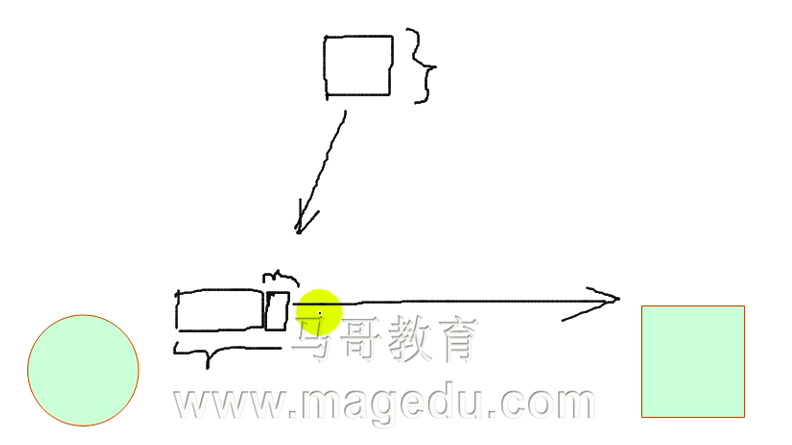
DSA仅仅支持身份认证 所以 只能是自己的私钥加密

公钥加密的加密数据太慢 ---- 所以通常用对方的公钥加密发送给对方的用来加密数据的对称密钥 ----- 所以 主要用来实现密钥交换 或者 加密特征码

私钥加密 也是加密数据的特征码



什么叫数字签名：就是让对方验证我们的身份的 ---- 身份认证就是用私钥加密我们的hash码 ----- 用私钥加密的特征码 就叫数字签名



单向加密算法 抽取特征码 然后私钥加密这个特征码 获取数字签名

这样 加密的结果 都可能被损坏 –发送给Alice Alice计算结果不一样 ---- 就会丢弃这段数据

对于Eve来讲 就没有办法了 ---保证了身份认证 和 数据完整性

但是Eve玩另一种手段 ----- 使劲改这个数据 这样Alice就不停地让bob重发 但是没有办法 Eve就是获取不到 也不让你获取到正确的信息 --- 这种篡改攻击无法做到

这时候 对原文数据对称加密不行

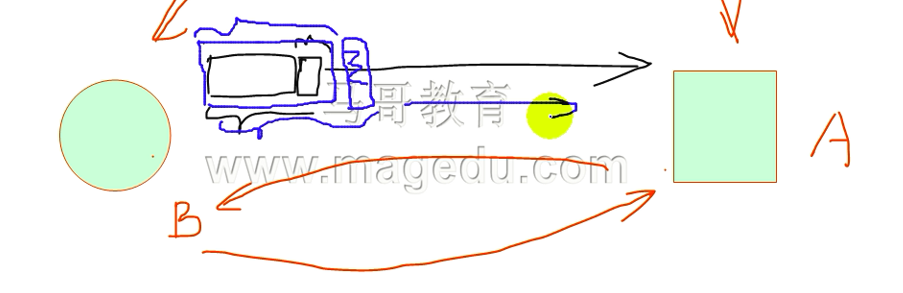
CA认证之后：

因此Alice每一次和Bob通信之前 要跟Bob索要证书 这样 就能获取对方的公钥 反之亦然

Bob要想和Alice通信 Alice就把证书发给Bob

这样 每次通信之前 双方先交换一下数据 bob私钥加密一段数据 发给Alice ----- Alice通过CA知道这个就是Bob ---- 这个就是A认证B的过程

Bob想把整个数据加密 不让任何第三方知道 就会把整个的数据都进行对称加密 对称加密的密钥使用Alice的公钥进行加密



这个时候 必须保证Alice的公钥就是Alice发送过来的 ---- 此时就必须使用CA的证书 来认证 认证通过 获取Alice的公钥

这个时候 CA在这种场景下诞生的 ---就是要验证每一个人就是他所声称的那个人

Alice如何认证这个证书是OK的？

 ---- 注意 这个证书的尾部有CA的签名 ----

Bob如何验证是Alice？首先Bob要获取到给Alice发证的CA的公钥 去解密这个签名 能解密的话 说明这个就是CA的签名 ---- 如果我认可这个CA 那么这个签名中的公钥就是可以认为是Alice的公钥 --- 用户名也可以认为是真的

如果Alice的私钥丢掉了 任何人都可能冒名去Alice 这个时候 Alice要吊销这个证书

所以Bob要解密+查询吊销列表

证书颁发机构+证书吊销列表 就构成了我们的PKI ---- 公钥基础设施

还有一个问题：

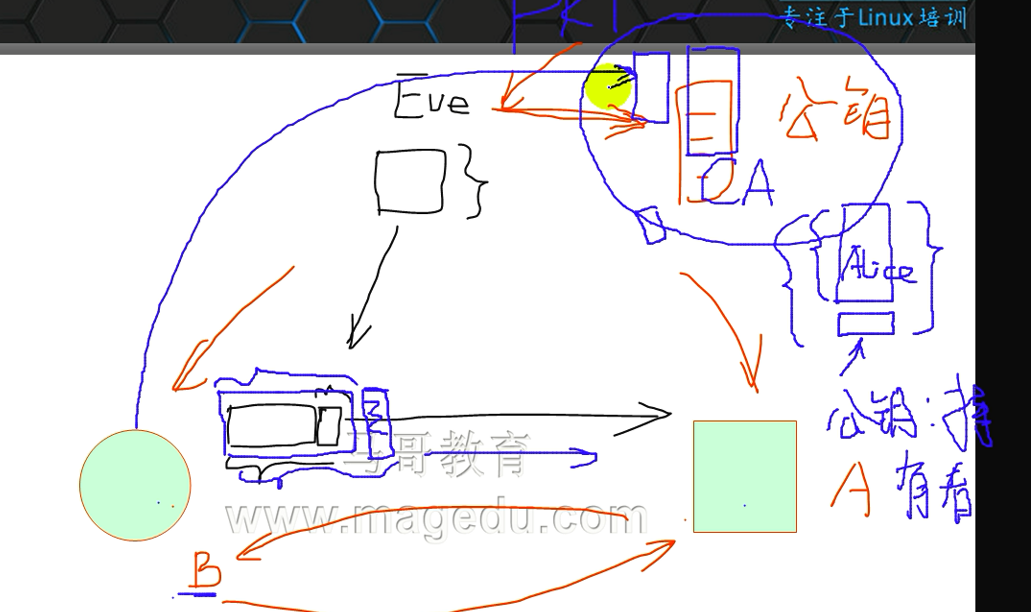
Bob要验证CA的证书 --- Bob要持有这个Alice的证书颁发者机构的公钥

如果此前Bob从来没有去过CA的网站下载公钥 如何确定这个网址就是CA的网站呢？ ---- 无从得知

所以 windows就会内置了公开的著名的CA

想要使用开源软件构建私有CA --- 比如建行 工行 如果买证书 那就太贵了

所以 可以用openssl或者openCA



----

问Eve此时只能望而兴叹 bob和Alice只能秘密通信

Eve此时 可以复制一份数据 发给Alice 一秒钟发送100w次 ------ 这种攻击叫做重放攻击

避免这种攻击 怎么办？有些软件可以阻止重放攻击

可以校验这个数据是否一致多次 ---拒绝服务攻击

什么是SSL？

应用场景：

假设Web服务器允许很多客户端来获取资源

现在假设是mail服务器

现在期望用户获取到的数据是私密的

但是http协议是明文的 数据发送的明文的 ---- 这样对于处心积虑的 就在交换机上面 复制一份数据 ---- 做一下解码 数据就被还原 ---- 所有数据 都经过交换机

如果不想要被看到 底层传输的报文 都是以太网的帧 --- ip报文是不加密的 tcp报文也不加密 http报文也不加密

没有加密的地方 这样想解码数据非常简单 ---- 使用简单的解码工具就能达到目的

同时 Linux本身有非常多的抓包解包工具

由此必须引入一种机制 本身不具备加密解密的协议 能够以加密解密的形式进行通信 ---- 比如在电子商务站点 或者网上银行 输入用户名和密码 不希望被别人看到

---- 如果可以随意截取 会非常麻烦

避免这种情况 https 说白了 就是引入了SSL/TLS的http协议

如何工作的呢？

Http是应用层 -> 传输层 -> 网络层 -> 帧 -> 物理层 这里面都没有加密

在应用层里面 加上半层加密 --- 这个半层就是负责加密之后的安全连接 【用半层的原因 就是 有些可能不需要加密 所以 一半加密 一半不加密】

客户端和服务端都需要这样的半层加密

SSL是一种协议 ----早期网景公司netscape 所提供的