|  |  |
| --- | --- |
| **计算机网络安全实验报告** | |
| OPENSSL实验 | |
| 图片包含 标牌  已生成极高可信度的说明 | |
| **姓名** | 田丰瑞 |
| **班级** | 软件73班 |
| **学号** | 2172213528 |
| **电话** | 18744296191 |
| **Email** | [tianfr@stu.xjtu.edu.cn](mailto:tianfr@stu.xjtu.edu.cn?subject=软件73%20田丰瑞) |
| **日期** | 2020-5-12 |

# 声明

网络安全实验的所有代码已经全部开源于我的GitHub（仅开源代码部分），项目地址：

<https://github.com/tianfr/Internet-Security-ExpCode/>

# openssl实验

## 实验要求

实现openSSL的签发证书功能。

## OPENSSL简介

### openssl简介

**openssl**是一个安全套接字层密码库，囊括主要的密码算法、常用密钥、证书封装管理功能及实现ssl协议。OpenSSL整个软件包大概可以分成三个主要的功能部分：SSL协议库libssl、应用程序命令工具以及密码算法库libcrypto。

**SSL**：Secure Socket Layer，安全套接字层协议，分为SSLv2和SSLv3两个版本，TSL在SSL3.0基础之上提出的安全通信标准化版。主要是为了加密传输数据而产生的协议，能使用户/服务器应用之间的通信不被攻击者窃听，并且始终对服务器进行认证，还可选择对用户进行认证。

SSL协议建立在可靠的传输层协议（TCP,UDP,SCTP）之上，应用层协议（HTTP,FTP,TELNRET）在SSL之上，SSL协议在应用层协议通信之前就已经完成了加密算法、服务器认证等工作。http协议调用了ssl协议,那么他就变成了https(http –>ssl–>https)。主要功能两个：

1. 加密解密在网络中传输的数据包，同时保护这些数据不被修改和伪造。

应用层——>ssl解密——>传输层

传输层——>ssl加密认证——>应用层

1. 验证网络对话中双方的身份。

**SSL握手协议**：SSL包含两个子协议，一个是包协议，一个是子协议。**包协议**说明SSL的数据包是如何封装的；**握手协议**说明通信双方协商通信双方决定使用什么算法及算法使用的key。下面事握手协议过程：

1. Client向服务器发送自己支持的协议版本（如TLS1.2）、client生成的随机数、自己加密算法的一些配置。
2. Server 收到Client请求后向客户端response:确认使用加密通信协议的版本、生成一个随机数、确认使用加密的方法、server certificate（服务器证书）。
3. Client 验证服务器证书，在确认无误后取出其公钥，并发送随机数 Pre-Master-Key（用于公钥加密）、编码变更通知（通信双方都用商定好的密钥进行通信;即随后的信息都将用双方商定好的加密方法和密钥发送. ）
4. Server验证完client的身份之后，用自己的私有密钥解密得到pre-master-key后,然后双方利用这个pre-master key来共同协商，得到master secret。返回信息给client
5. 双方用master一起产生真正的session key，这就是他们在剩下的过程中的对称加密的key了。这个key还可以用来验证数据完整性。双方再交换结束信息。握手结束。

**证书Certificate和证书颁发机构CA（certification authority）**

证书是建立公共密钥和某个实体之间联系的数字化的文件。它包含的内容有：版本信息（X.509也是有三个版本的）、系列号、证书接受者名称、颁发者名称、证书有效期、公共密钥、一大堆的可选的其他信息、CA的数字签名。证书由CA颁发，由CA决定该证书的有效期，由该CA签名。每个证书都有唯一的系列号。证书的系列号和证书颁发者来决定某证书的唯一身份。常用的证书是采用X.509结构的，这是一个国际标准证书结构，

CA是第三方机构，被你信任，由它保证证书的确发给了应该得到该证书的人。CA自己有一个庞大的public key数据库，用来颁发给不同的实体。CA也是一个实体，它也有自己的公共密钥和私有密钥。

**加密算法**

1）对称加密：指加密和解密使用相同密钥的加密算法。对称加密算法的优点在于加解密的高速度和使用长密钥时的难破解性

常见的对称加密算法：DES、3DES、DESX、AES、RC4、RC5、RC6等

2）非对称加密：指加密和解密使用不同密钥的加密算法，也称为公私钥加密

常见的非对称加密算法：RSA、DSA（数字签名用）等

3）Hash算法：Hash算法它是一种单向算法，用户可以通过Hash算法对目标信息生成一段特定长度的唯一的Hash值，却不能通过这个Hash值逆向获得目标信息

常见的Hash算法：MD2、MD4、MD5、SHA、SHA-1等

## OPENSSL实验环境配置

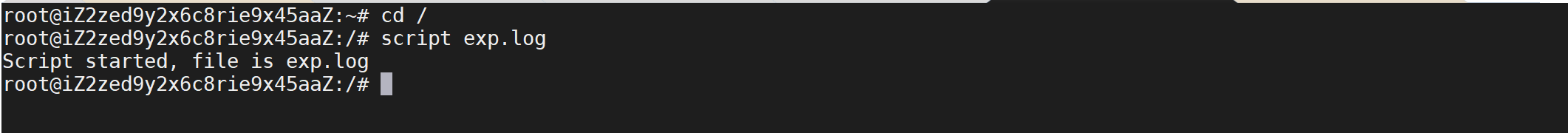
本次实验使用ubuntu16.04环境，并通过MobaXterm远程连接服务器，实验利用screen程序保证网络连接突然中断时的稳定性，并通过script程序将实验log本地保存。

下面配置实验环境截图：

建立screen会话：



保存实验会话记录：



# 实验过程

## 使用openssl生成RSA密钥对

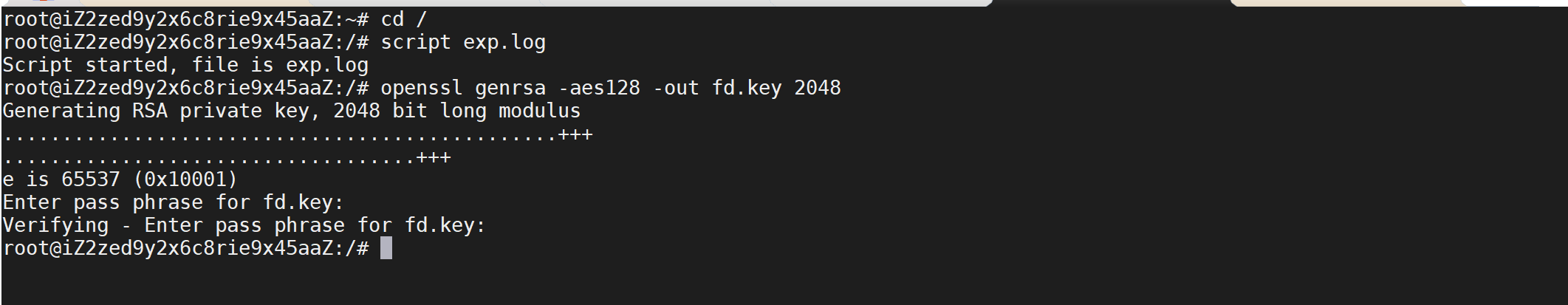
使用openssl的私钥产生公钥前，需要了解以下几点：

* + **key算法**：openssl 支持生成RSA，DSA，ECDSA的密钥对，但是RSA是目前使用最普遍的。
  + **Key长度**：RSA的2048是公认较比较安全的key长度。
  + **密码（Passphrase**）：在key上使用密码是一个可选值，但是一般都是强烈建议的（官网这样写的，实际项目中很多都没有设置口令），这样每次使用key文件时，都需要输入这个密码才能使用，增强了其安全性，但是随之而来的易用性也会变差。

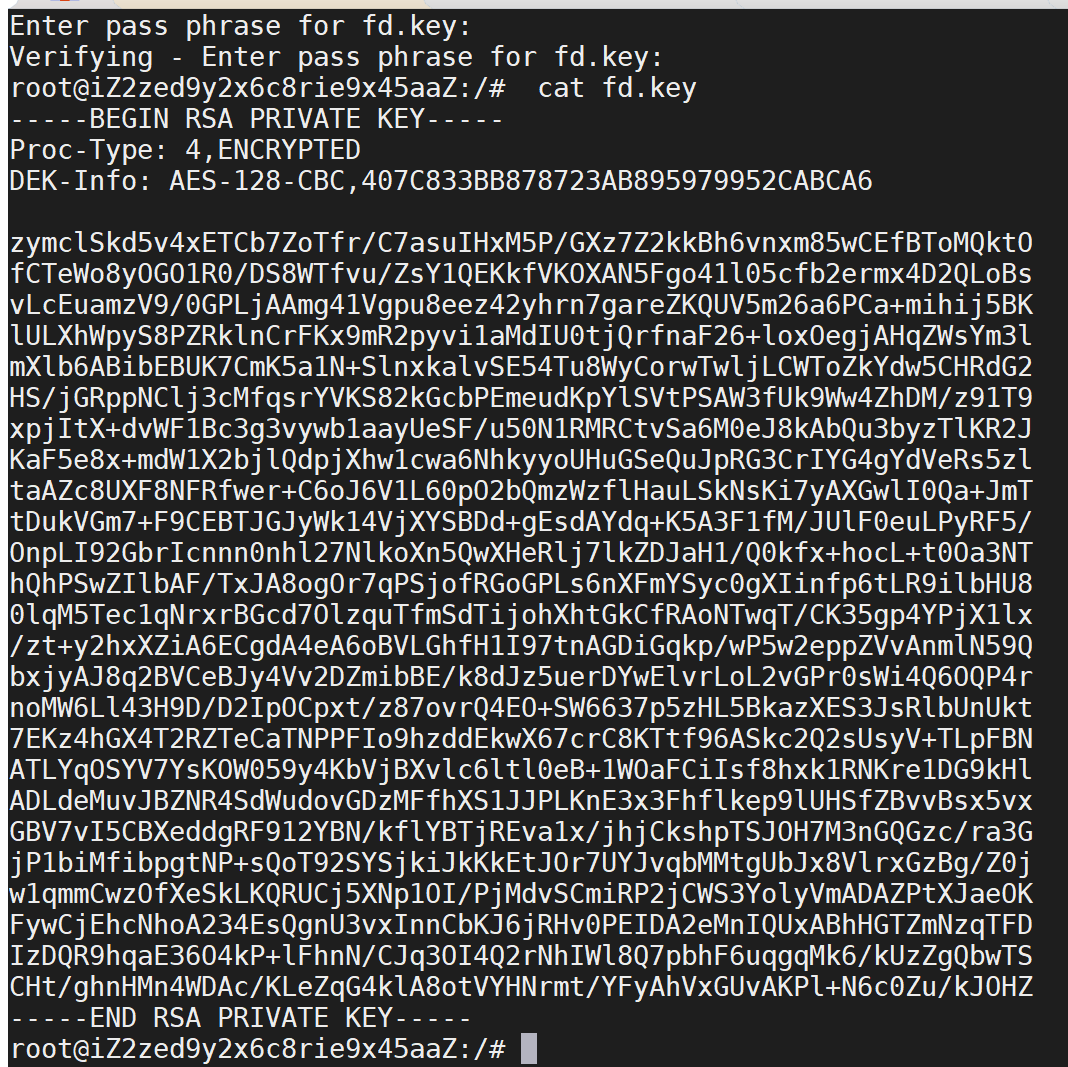
使用genrsa命令来生成RSA key。

### 生成私钥

**使用命令：openssl genrsa -aes128 -out fd.key 2048**。以下输入了为这个key值设置了密码，且密码使用aes128加密保存。

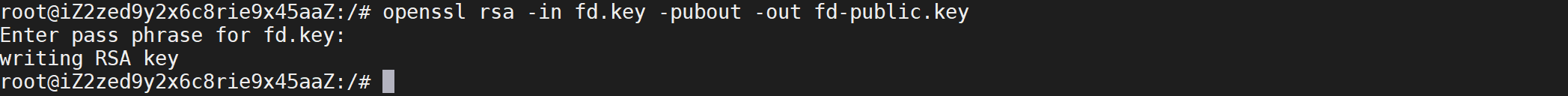


这个key文件就是私钥文件。利用cat命令可以查看下文件内容：

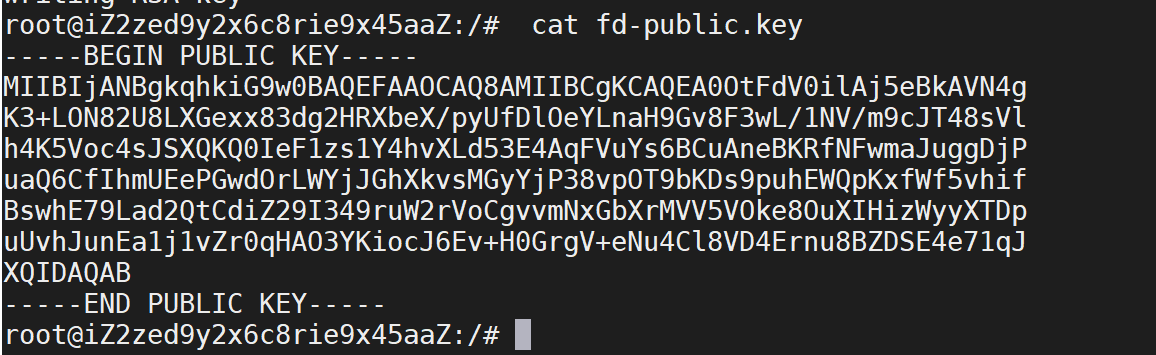


### 生成公钥

**使用命令：openssl rsa -in fd.key -pubout -out fd-public.key**



利用cat查看这个key文件，就是公钥：

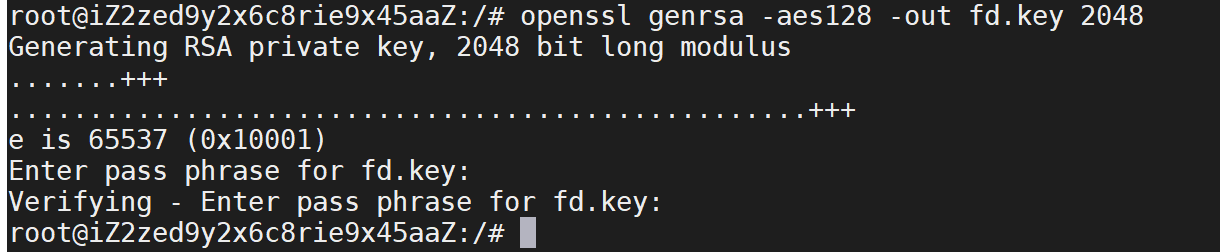


## 获取权威机构颁发证书

获取权威机构颁发的证书，需要先得到私钥的key文件（.key），然后使用私钥的key文件生成sign req 文件（.csr），最后把csr文件发给权威机构，等待权威机构认证，认证成功后，会返回证书文件（.crt）。

### 生成私钥key

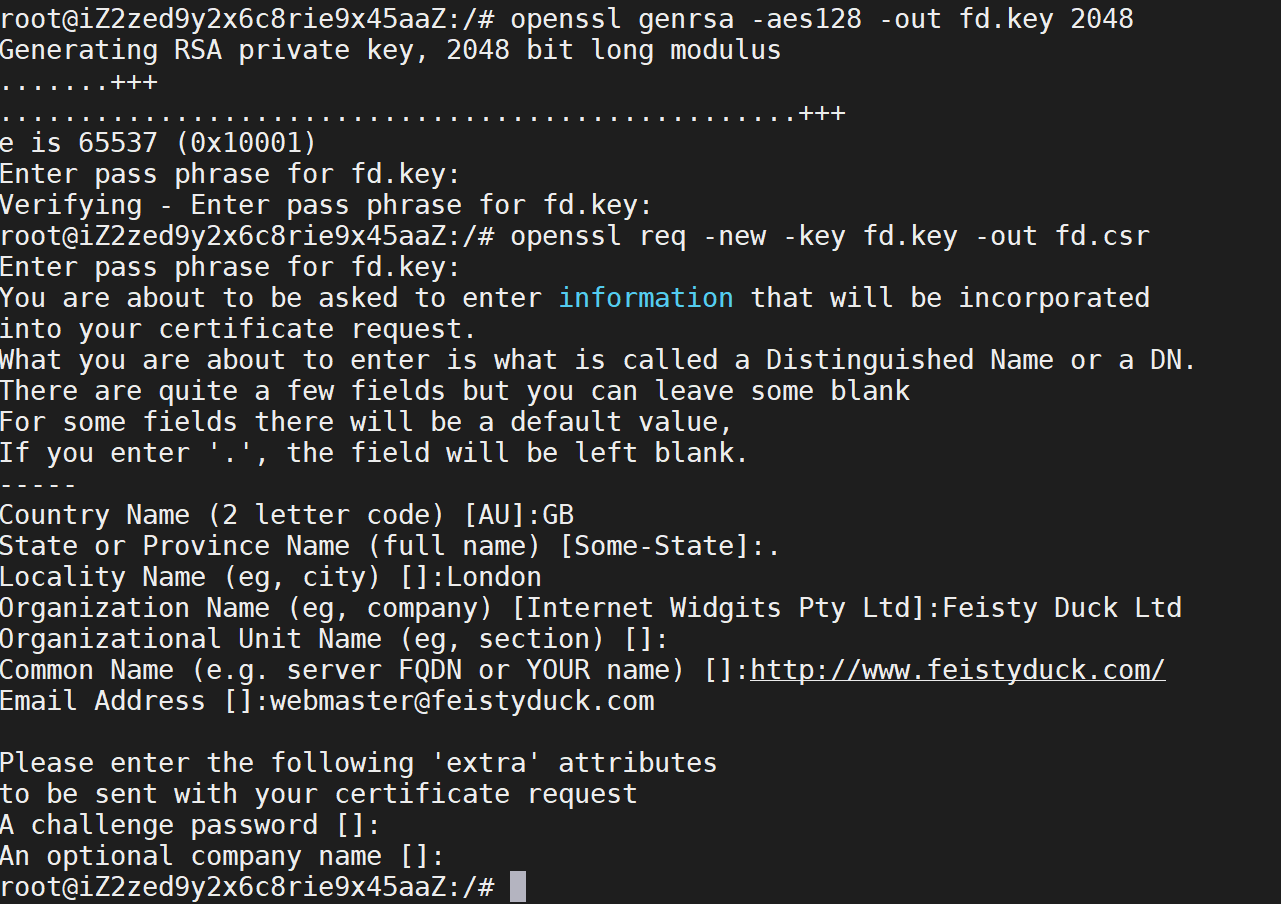
与第二节使用openssl生成RSA密钥对的步骤A一致。使用命令：openssl genrsa -aes128 -out fd.key 2048



### 私钥的key文件生成sign req 文件（.csr）

生成csr文件时，需要填写一些关于待签人或者公司的一些信息，比如国家名，省份名，组织机构名，主机名，email名，有些信息可以不填写，使用.标识。

**使用命令：openssl req -new -key fd.key -out fd.csr**。过程如下：



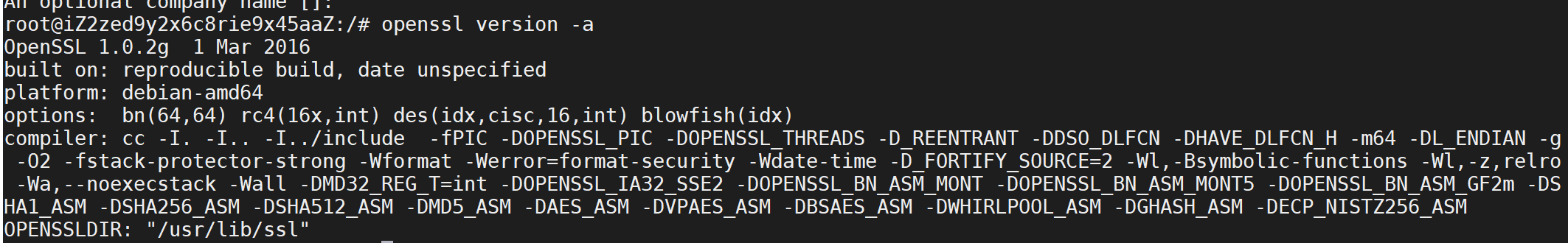
把csr文件发给权威机构，等待权威机构认证，交费获取证书即可。

## OpenSSL生成root CA及签发证书

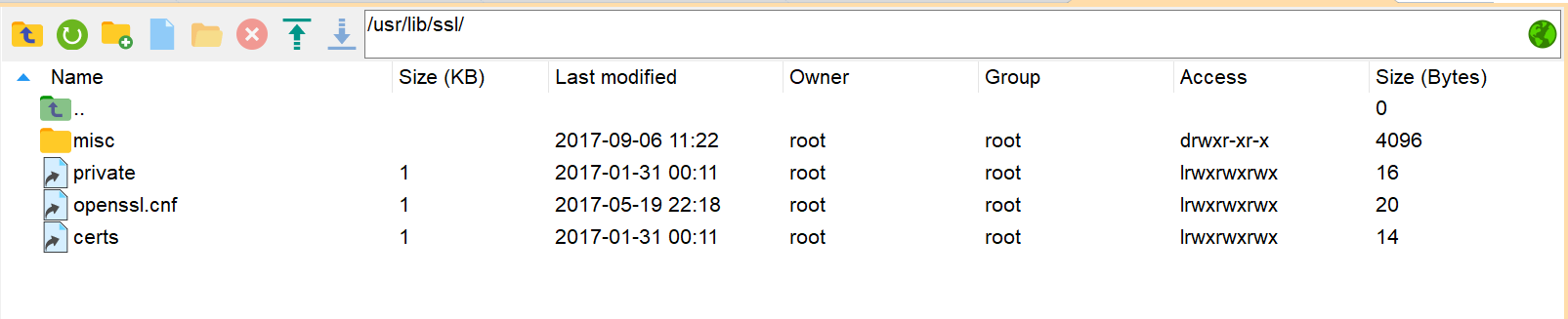
有时候，使用SSL协议是自己内部服务器使用的，这时可以不必去找第三方权威的CA机构做证书，可以做自签证书（自己创建root CA（非权威））主要有以下三个步骤。

### 创建openssl.cnf在使用default-ca时需要使用的SSL的工作目录

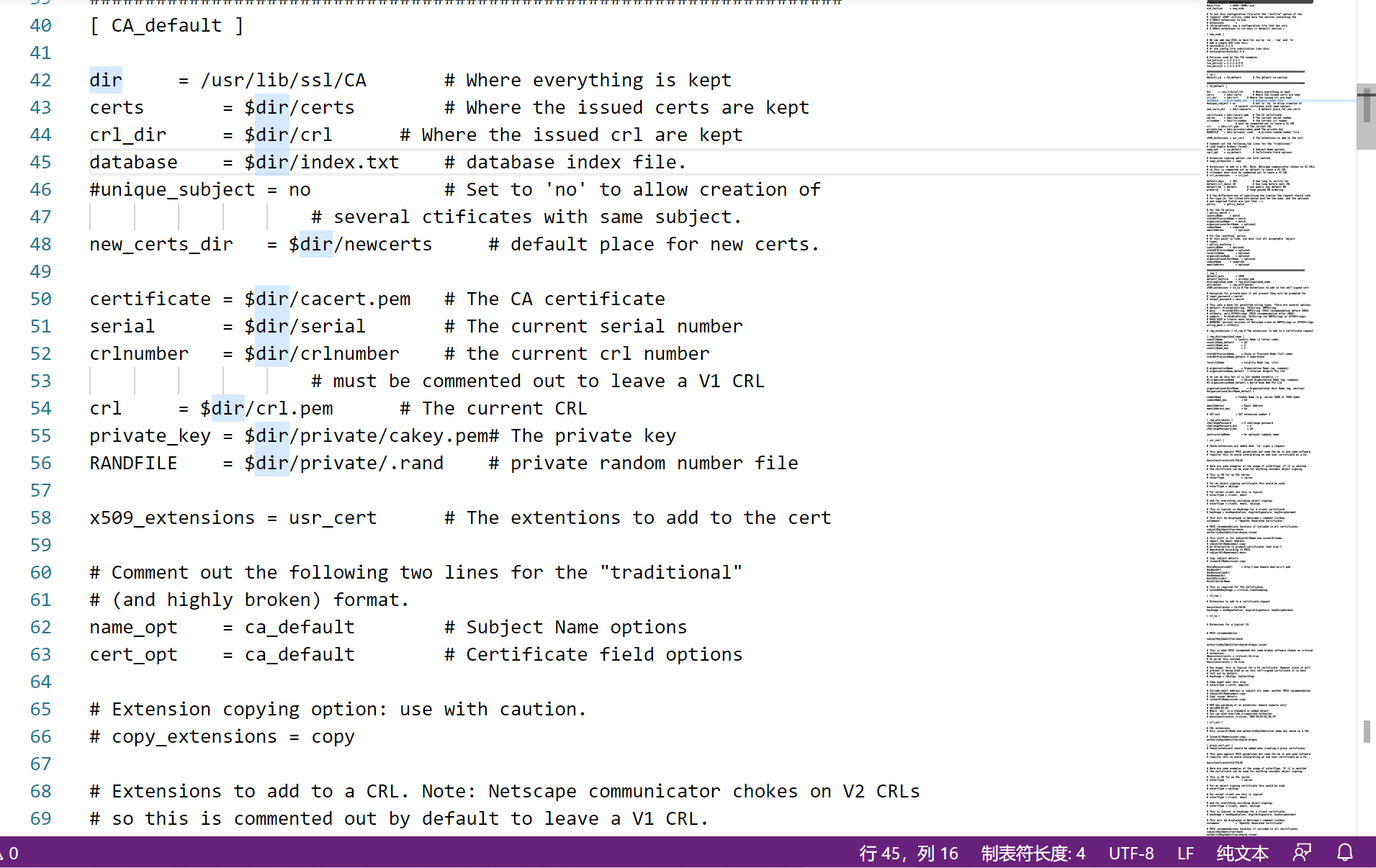
查看openssl的配置文件

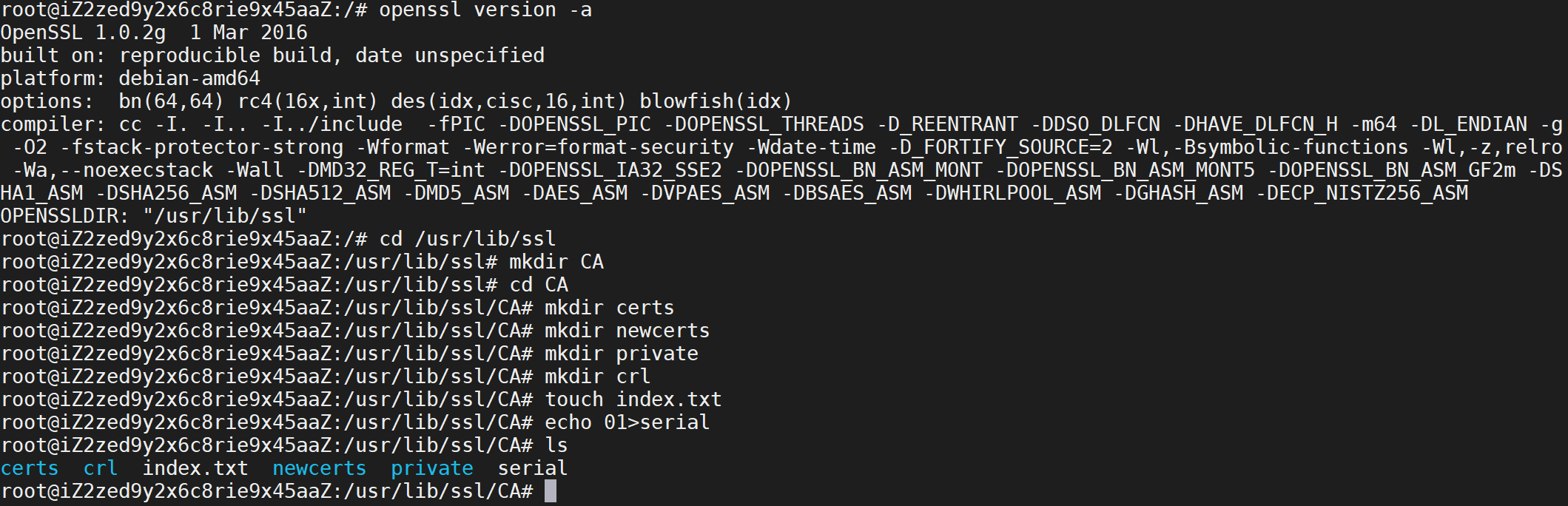


找到OPENSSLDIR: " /usr/lib/ssl/ "的配置文件openssl.cnf



根据配置文件下的[ CA\_default ]节点默认值，创建对应文件夹和文件





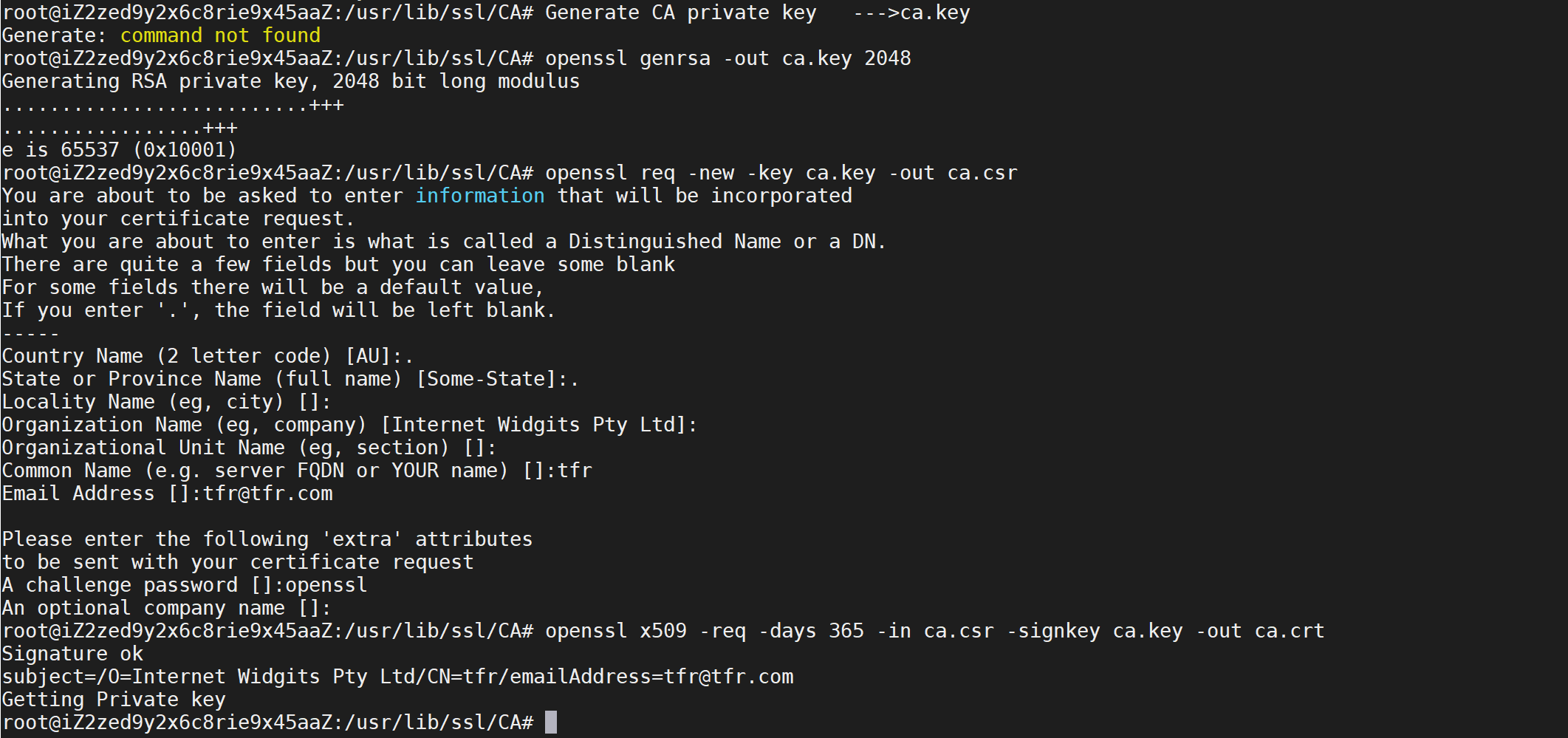
其中，certs：存放已颁发的证书；newcerts：存放CA指令生成的新证书；private：存放私钥；crl：存放已吊销的整数；index.txt：penSSL定义的已签发证书的文本数据库文件，这个文件通常在初始化的时候是空的；serial：证书签发时使用的序列号参考文件，该文件的序列号是以16进制格式进行存放的，该文件必须提供并且包含一个有效的序列号。

执行完后，当前目录为：



### 生成CA根证书（root ca证书）

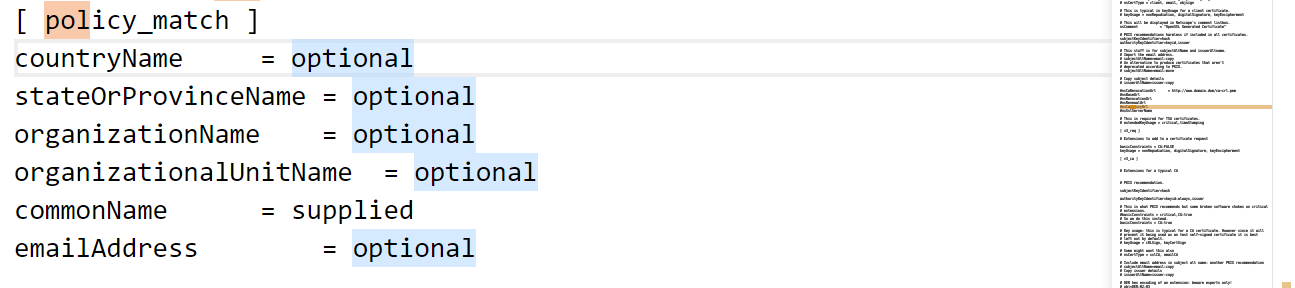
**步骤：生成CA私钥（.key）-->生成CA证书请求（.csr）-->自签名得到根证书（.crt）（CA给自已颁发的证书）。**

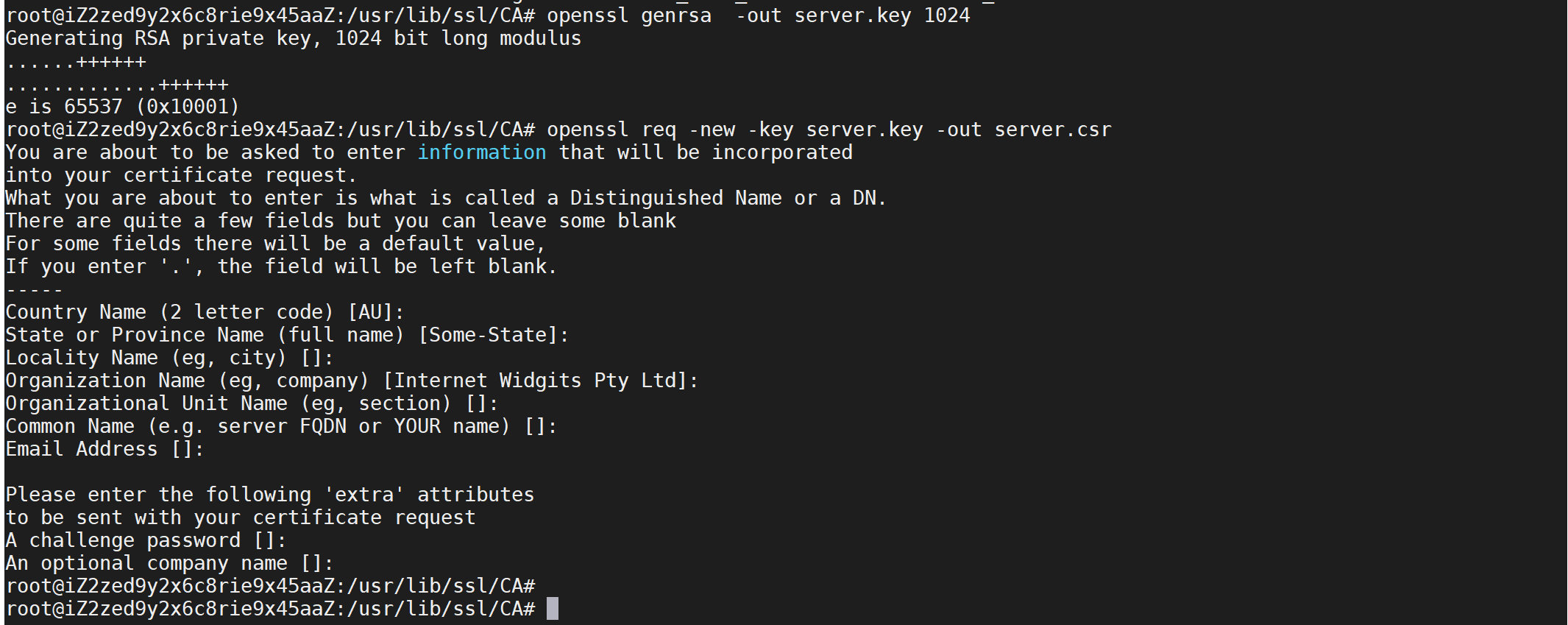


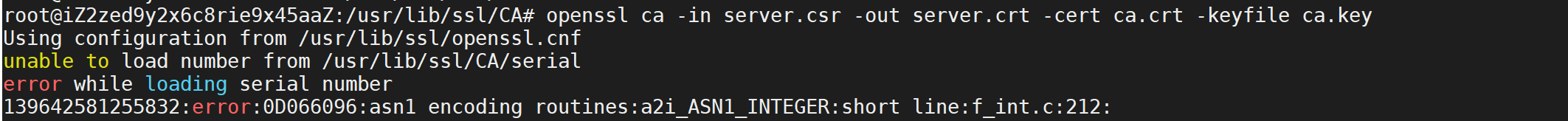
### **用自签根证书 ca.crt 给用户证书签名**

**步骤：生成私钥（.key）-->生成证书请求（.csr）-->用CA根证书签名得到证书（.crt）**

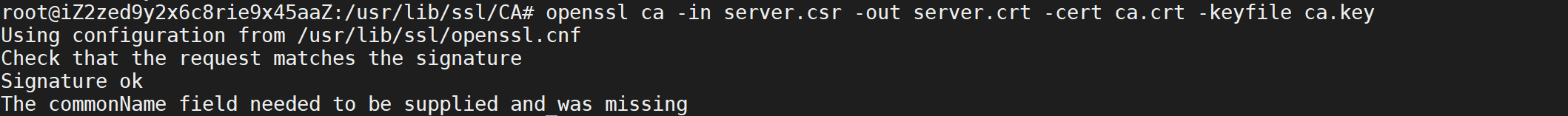
**修改openssl.cnf文件：**







发现出错，是ANSI编码问题。



至此openssl实验全部完成。

把server.crt以及server.key保存在服务器端等待程序加载使用；把ca.key保存在客户端，如果客户端需要验证服务器端发的证书时使用。

# 总结

本次实验加深了我对openssl的理解。Openssl可以用于网站传输加密。我此次实验让我进一步了解openssl的具体结构，了解了openssl在Ubuntu上的运行具体环境与机制。