**认证插件部分**

1. **实现原理**
   1. **服务器构成**

·初始化

SSLeay\_add\_ssl\_algorithms()

OpenSSL\_add\_all\_algorithms()

SSL\_load\_error\_strings()

ERR\_load\_BIO\_strings()

·等待接收到连接

Accept()

·使用GMTLS\_client\_method方法

SSL\_CTX\_new()

·加载CA的证书

SSL\_CTX\_load\_verify\_locations()

·加载自己的证书和私钥（包括证书和加密证书）

SSL\_CTX\_use\_certificate\_file()

SSL\_CTX\_use\_PrivateKey\_file()

私钥的作用是，ssl握手过程中，对客户端发送过来的随机消息进行加密，然后客户端再使用服务器的公钥进行解密，若解密后的原始消息跟客户端发送的消息一致，则认为此服务器是客户端想要链接的服务器

·验证私钥正确性

SSL\_CTX\_check\_private\_key()

·将连接交付给SSL并进行SSL连接

SSL\_new()

SSL\_connect()

·进行操作

·释放资源

SSL\_free()

SSL\_CTX\_free()

* 1. **客户端构成**

·初始化

·使用GMTLS\_client\_method方法

·要求校验服务器证书

SSL\_CTX\_set\_verify()

·加载CA证书

·加载自己的证书和私钥（包括证书和加密证书）

·创建连接、连接服务器

Socket()

Connect()

·将连接交付给SSL并进行SSL连接

·进行操作

·释放资源

* 1. **证书生成**

该部分在inership/mk\_tls\_cert/generate.sh中写的很详细，每一步都有说明这里仅给出gmssl req的部分参数信息

# gmssl req 创建证书签名请求等功能

# -nodes 对私钥不进行加密,不用手动输入密码

# -newkey 创建CSR证书签名文件和RSA私钥文件

# rsa:2048 指定创建的RSA私钥长度为2048

# -keyout 创建的私钥文件名称

# -out 指定CSR输出文件名

# -subj 指定证书Subject内容

# Subject设定内容说明

#

# 字段 含义 设定值例

# /C= Country CN

# /ST= State HUBEI

# /L= Location WUHAN

# /O= Organization HUST

# /OU= Organizational CA

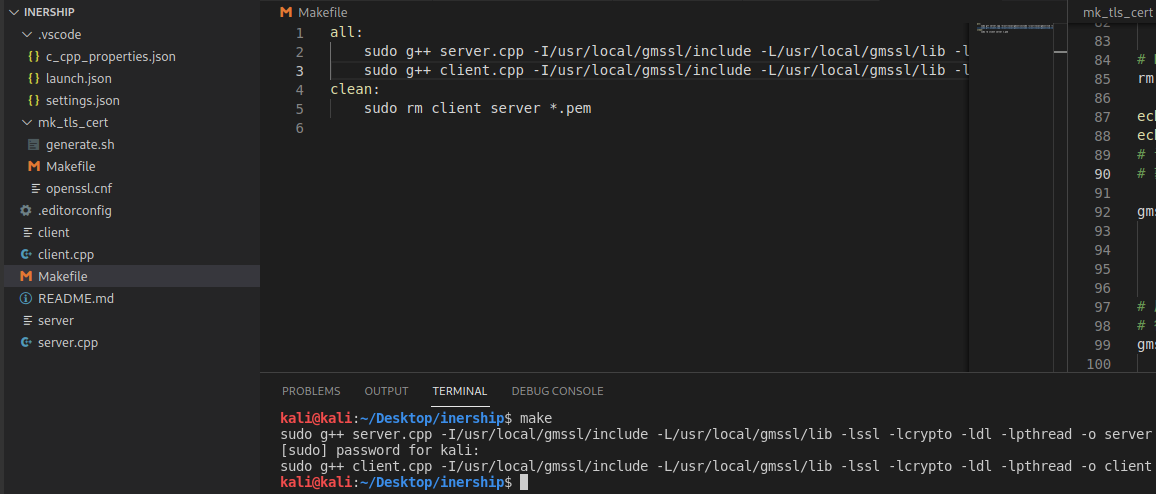
# /CN= Common Name sxy

1. **测试**

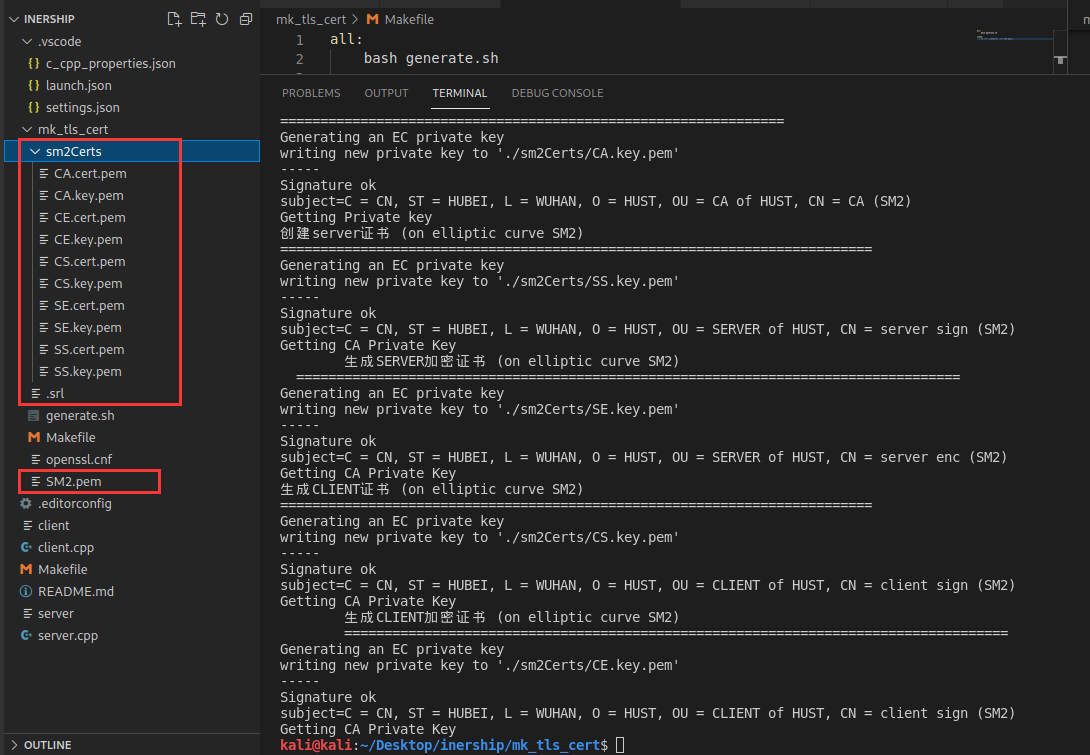
**环境：kali-linux-2020.1**

根据已经写好的Makefile文件，我们就可以简单的复现整个过程

首先编译server和client程序



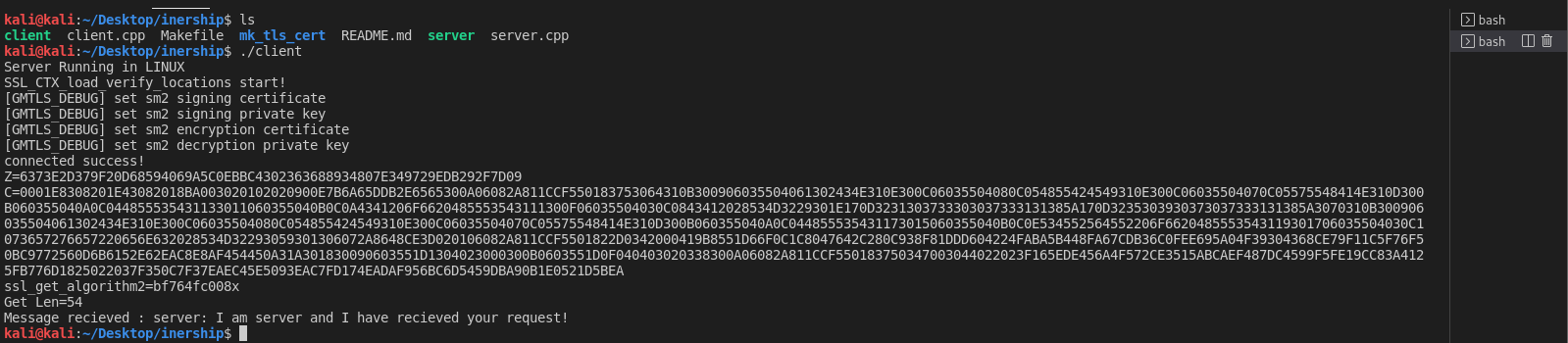
之后进入下级目录，同样使用makefile获得CA的证书与私钥、服务器的证书与私钥、服务器加密的证书和私钥、客户端的证书和私钥、客户端加密的证书和私钥



在一个terminal中执行server程序，显示在linux系统中执行，这里我用的是kali-linux



之后再开一个terminal执行client，可以看到”connected success!”，收到54个字符并显示字符串



回到server的terminal，同样可以看到类似情景，说明测试成功

