技 术 文 件

技术文件名称：openssl制作自签名证书及apache上的布署

技术文件编号：

版 本：

拟 制 吉茂锋E0006316

审 核

会 签

标准化

批 准

海宁奕斯伟集成电路设计有限公司

修改记录

| 文件编号 | 版本号 | 拟制人/  修改人 | 拟制/修改日期 | 更改理由 | 主要更改内容  （写要点即可） |
| --- | --- | --- | --- | --- | --- |
|  | 1.0 | 吉茂锋 | 2022-11-19 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| 注1：每次更改归档文件（指归档到事业部或公司档案室的文件）时，需填写此表。  注2：文件第一次归档时，“更改理由”、“主要更改内容”栏写“无”。 | | | | | |

目录

[一、基础](#_一、开发背景)概念1

1、[流媒体概念](#_1、流媒体概念)2

2、[HLS协议和原理](#_2、HLS协议和原理)5

3、[HLS 关键组成m3u8](#_3、HLS关键组成m3u8)7

[二、在线音乐的功能实现](#_二、音频框架中的在线音乐http拉取数据的具体实现)4

[1、http element的实现](#_1、http_element的实现) 5

[2、http Client的实现](#_2、http_Client的实现) 5

[3、http playlist 的实现](#_3、http_playlist_的实现) 5

[4、整体实现的大致流程图](#_4、整体实现的大致流程图) 5

# 一、开发背景及环境

前些日子，在我们的产品中（ecm9xx），使用mbedtls接口实现了https的通讯加密功能。当时开发时，调试使用的是外网测试网站，开发完成后，为了更好的验证ssl的功能，需要搭建了一个本地https的服务器。而https的服务器搭建，需要去一个CA机构去申请一个证书。但是CA机构的证书申请是要收费的，且还不便宜，所以作为测试用的https环境，就尽可能的用免费的证书了。因此，自己制作证书的需求就应运而生了。

我这里使用的环境及软件：

首先，我这里使用的操作系统是ubuntu18.04；

制作自签名证书的软件是OpenSSL软件；

搭建https的软件是apache（相对于nginx比较简单）。

# 1、自签名证书概述

因为自签名证书签发相对于商业证书流程简单，费用低廉（除了电费几乎不花钱），更新容易。所以在开发领域、甚至一些小众场景下特别常见，比如 K8S / MySQL 集群中的 TLS 认证，一些大的集团、公司的内网服务、网站安全证书、企业路由器设备的管理后台、用于管理企业员工的“安全准入客户端”等不乏使用这个方案。

经常有人说，使用自签名证书不安全，会导致中间人攻击。这里需要为自签名证书“正名”，如果你制作生成的证书被妥善保管（即不泄漏并被二次利用），并将其加入你的有限的设备（自用、团队使用）的证书信任列表中，在明确你的设备访问地址（不涉及DNS攻击），你是不会遇到中间人攻击的。

比如当你遇到类似下面的场景，不一定会遇到不安全的事情，有可能只是管理员忘记换掉过期证书、或者你自己生成证书后，使用了一台没有信任证书的设备进行访问、也可能是管理员压根没有想在公网签发证书，想做一个私有的网站：



一旦我们正确生成证书，在妥善保存证书后，进行了有限设备的白名单设置后，我们的证书和商业证书的使用是几乎没有差别的

# 2、Apache软件

## 2.1、Apache软件的安装

在ubuntu上安装的是apache2，安装命令如下：

1.sudo apt-get update

2.sudo apt-get install apache2

3.sudo apt-get remove apache2 #卸载

apache2安装成功后，我们可以在/var/www/目录下看到一个index.html文件，我们只需要重启apache2服务，使用同一局域网的设备登录ubuntu的ip地址就能访问到此页面了（http://ip/index.html）

## 2.2、重启apache2

重启apache2命令：

sudo /etc/init.d/apache2 restart

## 2.3、开启apache2的ssl访问功能

sudo a2enmod ssl #开启ssl模块

sudo a2dismod ssl #关闭ssl模块

## 2.4、ssl的配置修改，用来指向服务器的证书

ssl的配置修改，用来指向服务器的证书：

jimaofeng@X230:/etc/apache2/sites-available$ ls

000-default.conf default-ssl.conf

jimaofeng@X230:/etc/apache2/sites-available$ pwd

/etc/apache2/sites-available

jimaofeng@X230:/etc/apache2/sites-available$ vi default-ssl.conf

SSLCertificateFile /home/jimaofeng/demoCA/certs/server.srt

SSLCertificateKeyFile /home/jimaofeng/demoCA/private/server-key.pem

## 2.5、sites-available和sites-enabled目录

两者区别:

sites-available目录是存放可用的内容，但不起作用，只有用ln 连到sites-enabled目录才可以起作用。sites-enabled目录存放真正起作用的配置文件，存放一些指向sites-available目录的符号链接。所以apache上配置了多个虚拟主机，每个虚拟主机的配置都放在sites-available下，那么对于虚拟主机的停用和启用就非常方便。当sites-enabled下建立一个指向某个虚拟主机配置文件的连接时，就启用了它。如果要关闭某个虚拟主机的话，只需要删除相应的符号链接即可，不用去改配置文件

# 3、OpenSSL软件

## 2.1、OpenSSL软件的安装

在ubuntu上安装的是openssl，安装命令如下：

1. sudo apt-get install openssl

2. locate openss.cnf

openssl安装成功后，其命令主要分为三大功能：对称加密、单向加密、证书管理。可用openssl help 查看它所有的命令

jimaofeng@X230:~$ openssl help

Standard commands

asn1parse ca ciphers cms

crl crl2pkcs7 dgst dhparam

dsa dsaparam ec ecparam

enc engine errstr gendsa

genpkey genrsa help list

nseq ocsp passwd pkcs12

pkcs7 pkcs8 pkey pkeyparam

pkeyutl prime rand rehash

req rsa rsautl s\_client

s\_server s\_time sess\_id smime

speed spkac srp storeutl

ts verify version x509

Message Digest commands (see the `dgst' command for more details)

blake2b512 blake2s256 gost md4

md5 rmd160 sha1 sha224

sha256 sha3-224 sha3-256 sha3-384

sha3-512 sha384 sha512 sha512-224

sha512-256 shake128 shake256 sm3

Cipher commands (see the `enc' command for more details)

aes-128-cbc aes-128-ecb aes-192-cbc aes-192-ecb

aes-256-cbc aes-256-ecb aria-128-cbc aria-128-cfb

aria-128-cfb1 aria-128-cfb8 aria-128-ctr aria-128-ecb

aria-128-ofb aria-192-cbc aria-192-cfb aria-192-cfb1

aria-192-cfb8 aria-192-ctr aria-192-ecb aria-192-ofb

aria-256-cbc aria-256-cfb aria-256-cfb1 aria-256-cfb8

aria-256-ctr aria-256-ecb aria-256-ofb base64

bf bf-cbc bf-cfb bf-ecb

bf-ofb camellia-128-cbc camellia-128-ecb camellia-192-cbc

camellia-192-ecb camellia-256-cbc camellia-256-ecb cast

cast-cbc cast5-cbc cast5-cfb cast5-ecb

cast5-ofb des des-cbc des-cfb

des-ecb des-ede des-ede-cbc des-ede-cfb

des-ede-ofb des-ede3 des-ede3-cbc des-ede3-cfb

des-ede3-ofb des-ofb des3 desx

rc2 rc2-40-cbc rc2-64-cbc rc2-cbc

rc2-cfb rc2-ecb rc2-ofb rc4

rc4-40 seed seed-cbc seed-cfb

seed-ecb seed-ofb sm4-cbc sm4-cfb

sm4-ctr sm4-ecb sm4-ofb

## 2.2、openssl 对称加密

对称加密需要使用的标准命令为 enc ，用法如下：

openssl enc -ciphername [-in filename] [-out filename] [-pass arg] [-e] [-d] [-a/-base64][-A] [-k password] [-kfile filename] [-K key] [-iv IV] [-S salt] [-salt] [-nosalt] [-z] [-md][-p] [-P] [-bufsize number] [-nopad] [-debug] [-none] [-engine id]

常用选项有：

-in filename：指定要加密的文件存放路径

-out filename：指定加密后的文件存放路径

-salt：自动插入一个随机数作为文件内容加密，默认选项

-e：可以指明一种加密算法，若不指的话将使用默认加密算法

-d：解密，解密时也可以指定算法，若不指定则使用默认算法，但一定要与加密时的算法一致

-a/-base64：使用-base64位编码格式

## 2.2、openssl 单向加密

dgst是openssl单向加密的命令，其用法为：

openssl dgst [-md5|-md4|-md2|-sha1|-sha|-mdc2|-ripemd160|-dss1] [-c] [-d] [-hex] [-binary] [-out filename] [-sign filename] [-keyform arg] [-passin arg] [-verify filename] [-prverify filename] [-signature filename] [-hmac key] [file…]

其中常用选项有

[-md5|-md4|-md2|-sha1|-sha|-mdc2|-ripemd160|-dss1]：指定一种单向加密算法；

-out FILENAME：将加密的内容保存到指定的文件中；

OpenSSL还支持生成密码的hash离散值，其子命令为passwd，语法如下：

openssl passwd [-crypt] [-1] [-apr1] [-salt string] [-in file] [-stdin] [-noverify] [-quiet] [-table] {password}

常用选项为：

-salt STRING：添加随机数；

-in FILE：对输入的文件内容进行加密；

-stdin：对标准输入的内容进行加密；

示例：

jimaofeng@X230:~/workspace$ openssl md5 src.tar

MD5(src.tar)= 92eb9694987b5183ae79fddc0b308263

jimaofeng@X230:~/workspace$ md5sum src.tar

92eb9694987b5183ae79fddc0b308263 src.tar

jimaofeng@X230:~/workspace$

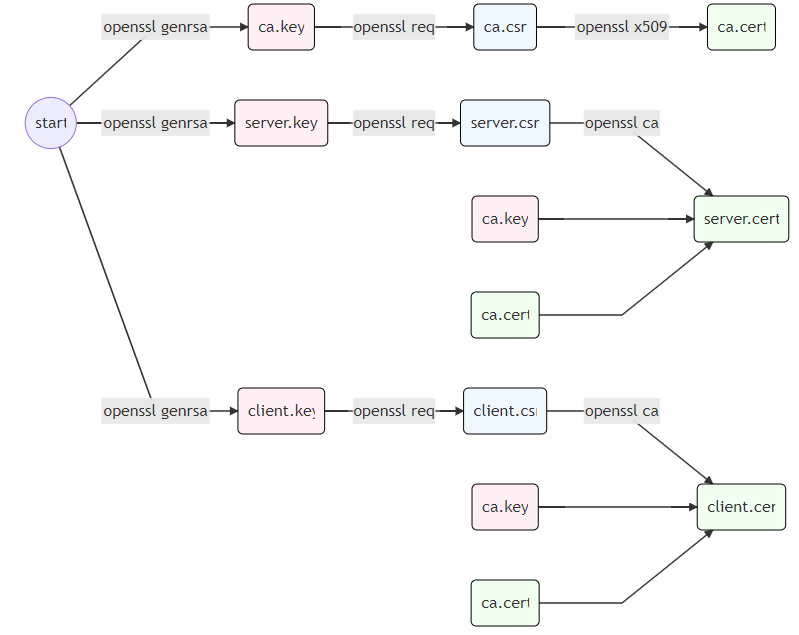
jimaofeng@X230:~/workspace$ openssl dgst src.tar

SHA256(src.tar)= 977d1dbe719af474ae1d071ed483c5baf4d846d2e33a30e882eb6e3a05389bac

jimaofeng@X230:~/workspace$

# 二、自签名证书制作详细步骤

根据网上查找的信息，以及自己本地的操作验证，特整理了用openssl制作自签名证书的步骤。



下面分步详细介绍

# 1、第一步，先修改openssl的配置文件 openssl.cnf

jimaofeng@X230:~/demoCA/certs$ vi /etc/ssl/openssl.cnf

[ req ]

default\_bits = 2048

default\_keyfile = privkey.pem

distinguished\_name = req\_distinguished\_name

attributes = req\_attributes

x509\_extensions = v3\_ca # The extensions to add to the self signed cert

string\_mask = utf8only

req\_extensions = v3\_req # The extensions to add to a certificate request

[ req\_distinguished\_name ]

countryName = Country Name (2 letter code)

countryName\_default = CN #jmf

countryName\_min = 2

countryName\_max = 2

stateOrProvinceName = State or Province Name (full name)

stateOrProvinceName\_default = GUANGDONG #jmf

localityName = Locality Name (eg, city)

localityName\_default = SHENZHEN #jmf

0.organizationName = Organization Name (eg, company)

0.organizationName\_default = AiSiNi #Internet Widgits Pty Ltd

# we can do this but it is not needed normally :-)

#1.organizationName = Second Organization Name (eg, company)

#1.organizationName\_default = World Wide Web Pty Ltd

organizationalUnitName = Organizational Unit Name (eg, section)

organizationalUnitName\_default = SoftWare #jmf

commonName = jimaofeng #Common Name (e.g. server FQDN or YOUR name)

commonName\_max = 64

emailAddress = 123jmf@163.com #Email Address

emailAddress\_max = 64

# SET-ex3 = SET extension number 3

[ v3\_req ]

# Extensions to add to a certificate request

basicConstraints = CA:FALSE

#keyUsage = nonRepudiation, digitalSignature, keyEncipherment

subjectAltName = @alt\_names

[alt\_names]

IP.1 = 10.19.96.155

IP.2 = 127.0.0.1

[ v3\_ca ]

# Extensions for a typical CA

# PKIX recommendation.

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer

basicConstraints = critical,CA:true

[ crl\_ext ]

# 2、第二步，生成根证书

jimaofeng@X230:~/demoCA$ openssl genrsa -aes256 -out private/cakey.pem 2048

jimaofeng@X230:~/demoCA$ openssl req -config /etc/ssl/openssl.cnf -key private/cakey.pem -new -x509 -days 3650 -sha256 -extensions v3\_ca -out certs/cacert.pem

# 3、第三步，用根证书cacert来签名服务器证书

首先要先生成服务器的密钥server-key.pem

jimaofeng@X230:~/demoCA$ openssl genrsa -aes256 -out private/server-key.pem 2048

然后生成服务器的证书请求文件server.csr

jimaofeng@X230:~/demoCA$ openssl req -new -key private/server-key.pem -out private/server.csr -config /etc/ssl/openssl.cnf

用根证书来签名服务器证书请求文件server.csr，从而生成服务器的证书 server.srt

jimaofeng@X230:~/demoCA$ openssl x509 -req -days 365 -sha256 -extensions v3\_req -CA certs/cacert.pem -CAkey private/cakey.pem -CAserial ca.srl -CAcreateserial -in private/server.csr -out certs/server.srt

# 4、第四步，修改好 https的默认配置并重启apache2

jimaofeng@X230:/etc/apache2/sites-available$ vi default-ssl.conf

<IfModule mod\_ssl.c>

<VirtualHost \_default\_:443>

ServerAdmin jimaofeng

SSLCertificateFile /home/jimaofeng/demoCA/certs/server.srt

SSLCertificateKeyFile /home/jimaofeng/demoCA/private/server-key.pem

SSLCACertificatePath /home/jimaofeng/demoCA/certs/

SSLCACertificateFile /home/jimaofeng/demoCA/certs/cacert.pem

重启apache2:

jimaofeng@X230:/etc$ sudo /etc/init.d/apache2 restart

# 三、使用自签名证书搭建Apache的https服务器

到目前为止，我们已经制作用了根证书 certs/cacert.pem ，并用根证书签发了服务器server的证书 certs/server.srt

然后把 cacert.pem server.srt 复制到windows10上(D:\cer\san)，直接修改扩展名为 cacert.crt 然后开始导入到chrome浏览器中

打开chrome浏览器，设置chrome://settings/ -》隐私设置和安全性 -》安全--》管理证书--》在弹出的证书框中，选择“受信任的根证书颁发机构”--》导入 ，先择上面的 cacert.crt,然后下一步。。。

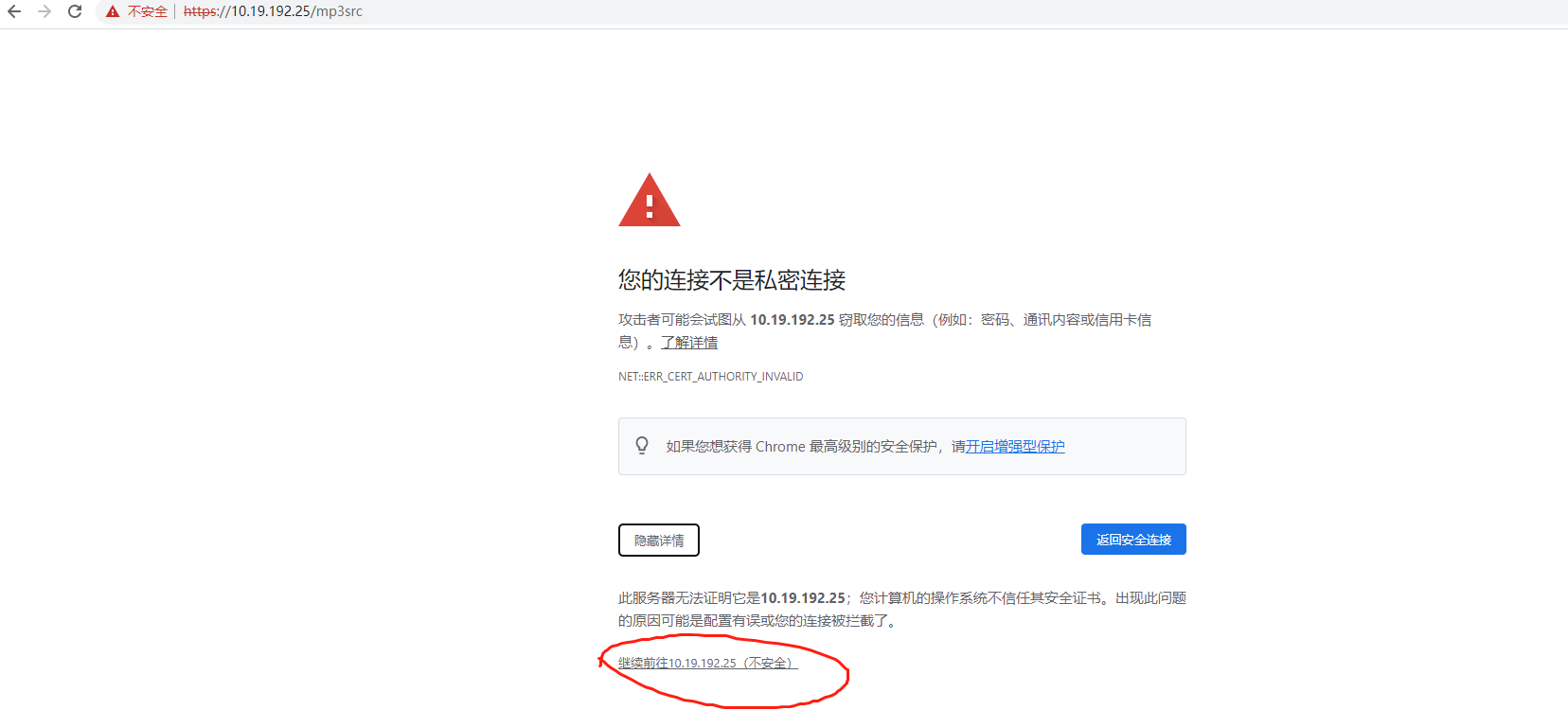
然后在地址栏中：输入 https://10.19.192.25/mp3src/

选择继续->下一步.

就显示出下面的东西了。

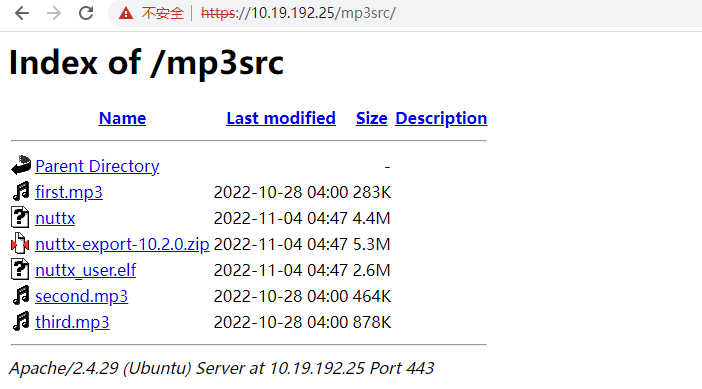


点开”高级”后，显示如下：



再点”继续前往…”,

就可以显出下列资源了。



# 四、参考文献

国外有个大牛写的一篇比较好的文章可以参考：

<https://jamielinux.com/docs/openssl-certificate-authority/index.html>

国内网址：

<https://blog.csdn.net/scuyxi/article/details/54884976>

<https://zhuanlan.zhihu.com/p/107741096>

https://blog.csdn.net/dreamback1987/article/details/103494304