**X.509解析**

**实验报告**

姓名：宋晓彤

学号：16340192

方向：嵌入式软件与系统

2018.12.12

# X.509证书结构

x.509标准规定了证书可以包含什么信息，并说明了记录信息的方法。

X.509结构中包括版本号（integer）、序列号（integer）、签名算法（object）、颁布者（set）、有效期（utc\_time）、主体（set）、主体公钥（bit\_string）、主体公钥算法（object）、签名值（bit\_string）。

使用ASN.1描述，我们可以将其抽象为以下结构

**Certificate::=SEQUENCE{**

**tbsCertificate TBSCertificate,**

**signatureAlgorithm AlgorithmIdentifier,**

**signatureValue BIT STRING**

**}**

**TBSCertificate::=SEQUENCE{**

**version [0] EXPLICIT Version DEFAULT v1,**

**serialNumber CertificateSerialNumber,**

**signature AlgorithmIdentifier,**

**issuer Name,**

**validity Validity,**

**subject Name,**

**subjectPublicKeyInfo SubjectPublicKeyInfo,**

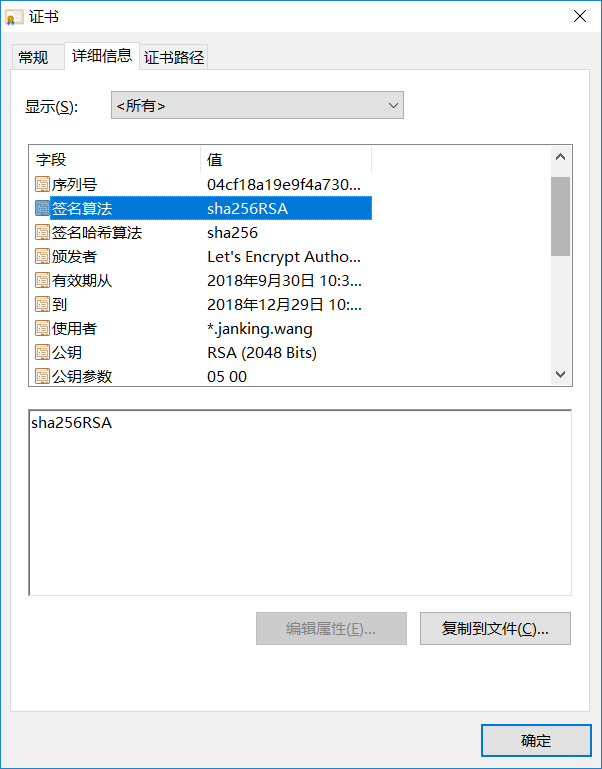
**issuerUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL,**

**subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL,**

**extensions [3] EXPLICIT Extensions OPTIONAL**

**}**

而本次实验，我选择使用从chrome上直接下载证书，此时我们可以看到，证书结构如下：



|  |  |  |  |
| --- | --- | --- | --- |
| 类 | 结构 | 信息 | 备注 |
| TBSCertificate | 版本信息 | 证书的使用版本 | 整数格式，0-V1，1-V2，2-V3 |
| TBSCertificate | 序列号 | 每个证书都有一个唯一的证书序列号 | 整数格式 |
| TBSCertificate | 签名算法 | 得到签名时使用的算法 | 有OID与之对应 |
| TBSCertificate | 颁发者 | 命名命规则一般采用X.500格式 | Name |
| TBSCertificate | 有效期 | 通用的证书一般采用UTC时间格式，计时范围为1950-2049 | Format：yymmddhhmssZ |
| TBSCertificate | 使用者 | 使用证书的主体 | Name |
| TBSCertificate | 主体密钥 | 证书所有人的公开密钥 |  |
| Certificate | 公钥签名算法 | 证书公钥的加密算法 | 有OID与之对应 |
| Certificate | 签名值 | 得到的签名结果 |  |

# 数据结构

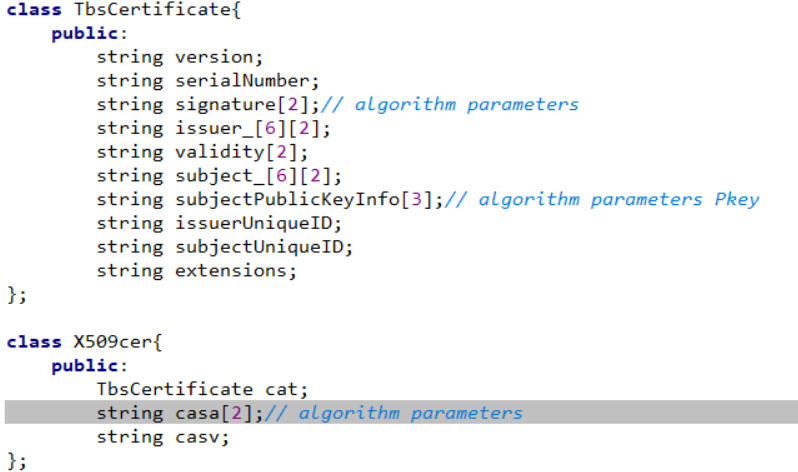
**【编码方法】**

X509的编码方法为TLV结构，使用T记录当前数据的类型（type），使用L记录当前数据的长度（length），使用V记录当前数据的取值（value），其中，不同的type值对应不同的数据类型

|  |  |  |
| --- | --- | --- |
| Type | 数据类型 | 编码格式 |
| 01 | Boolean | 01；01；FF/00 |
| 02 | Integer | 长度大于7f时，长度n与0x80进行“位或”运算的结果赋给length的第一个字节 |
| 03 | Bit string | 填充0成为8的倍数，Value的第一个字节记录填充数 |
| 04 | Ectet string | 04；len；val |
| 05 | Null | value部分为空，一共两字节 |
| 06 | Object Identifier | V1.V2.V3.V4.V5....Vn (1)计算40\*V1+V2作为第一字节；(2)将Vi(i>=3)表示为128进制，每一个128进制位作为一个字节，再将除最后一个字节外的所有字节的最高位置1；(3)依次排列，就得到了value部分 |
| 19 | ASCII string | 13；len；val |
| 23 | UTCtime | yymmddhhmssZ |
| 24 | Generalize time | yyyymmddhhmssZ |
| 48 | Sequence constructer | 序列内所有项目的编码的依次排列 |
| 49 | Set constructer | 集合内所有项目的编码 |
| 160 | Tag | 对于简单类型，type=80+tag序号；对于构造类型，type=A0+tag序号。length和value不变 |

**【数据结构】**

类的声明：均使用string类型记录数据，数据具体的内容已经在注释中标出



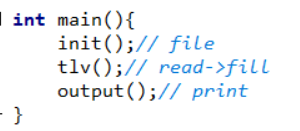
# 算法流程

1. 打开.cer文件，选择按字节读取，即每次读取一个字符
2. 读取的整体流程如下：
   1. 读取一个字节的type
   2. 读取一个字节的length
   3. 对type进行判断：如果type是非标签的：直接根据类型判断当前的数据类型是什么
   4. 对real length进行判断：根据type决定读取的数据长度，如integer区分长短数据，减去0x80后才是真正的长度值，同时换算出真正的长度
   5. 对value进行记录：根据长度读入实际的数据，并转换成自己需要的格式，如06的格式为V11.V2.V3….
3. 每次读取到value后，直接赋值给证书中的内容，此时，我们首先需要对当前的赋值对象进行判断
   1. 根据证书的结构，我们对读取的过程划分为以下阶段： "ver", "seq", "sigalg", "iss", "starttime","endtime", "usr", "keyalg",”sigalg”
   2. 此时，我们可以根据当前的赋值阶段n和此时读取的数据类型type来判断当前的赋值对象究竟是什么，避免产生因为发布者数目不统一而无法读取所有.cer文件的问题

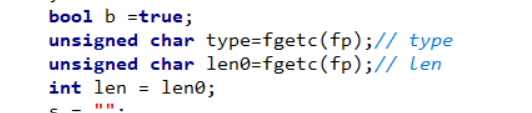
P.S, 参考了https://www.cnblogs.com/jiu0821/p/4598352.html并进行了很大程度上的优化

# 实现源码

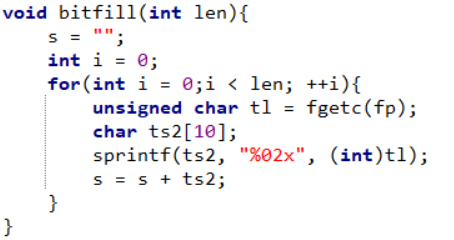
**【函数流程】**



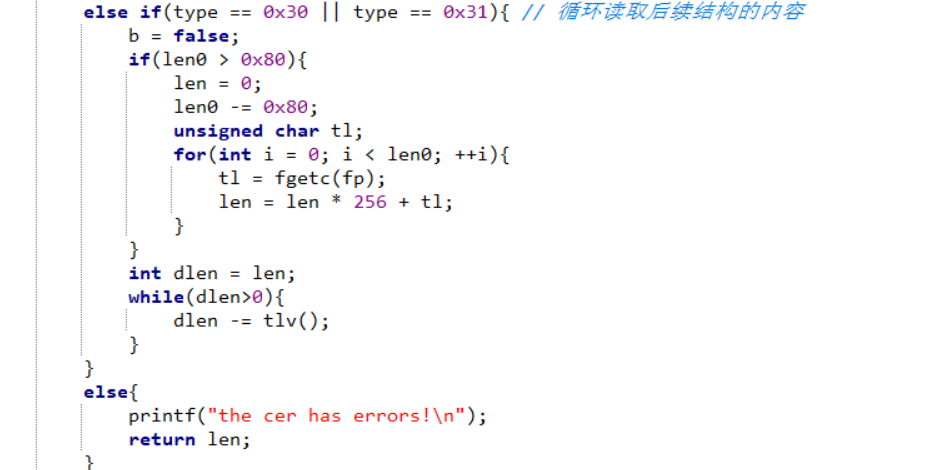
**【读取type和长度】**



**【根据长度读取数据】**

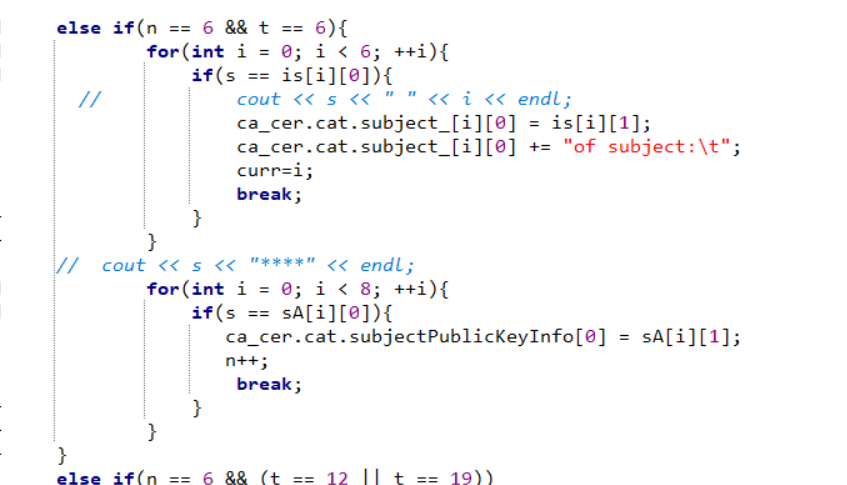


**【结构的循环读取】**



【根据n和type赋值】

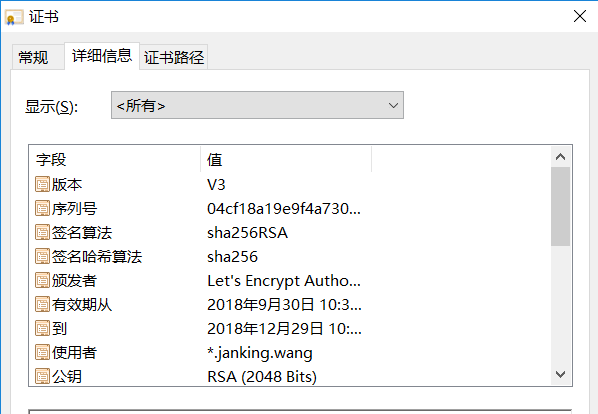
（在主体和主体公钥信息的读取上有冲突，但是可以一起解决）



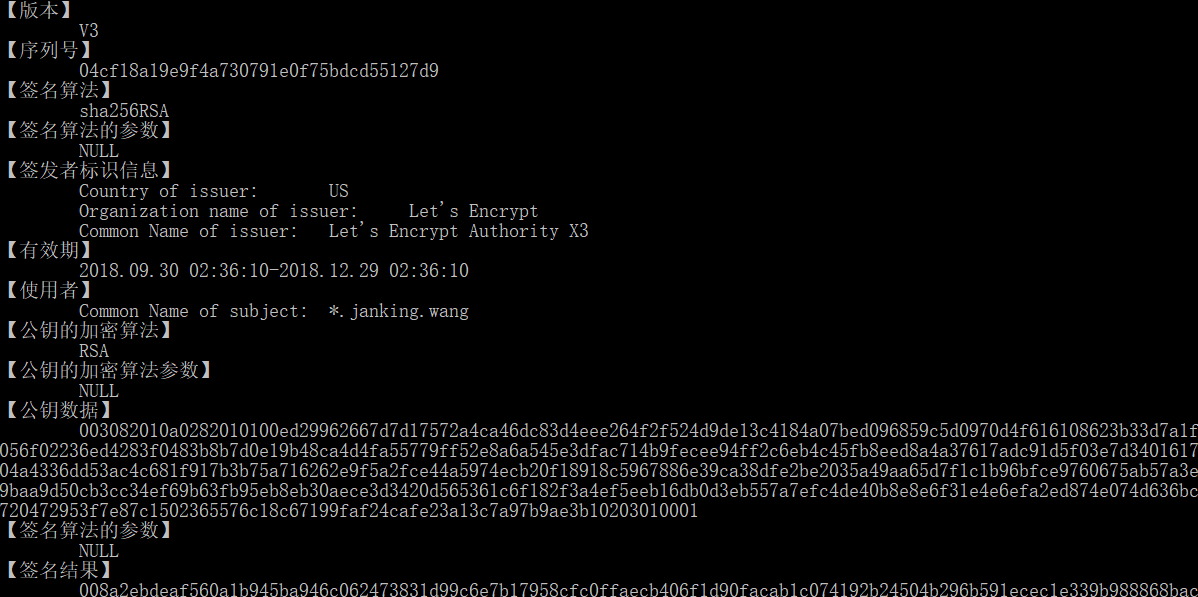
# 实验结果

进入chrome并打开一个网页，在安全标识处打开证书，查看内容，并下载DER编码格式的.cer文件，命名为test.cer

网页上直接查看结果如下



程序运行结果如下



**对照后发现，读取成功！**