

X.509解析程序 实验报告

16340063 巩泽群

一、实验结果

本次解析如下证书

证书内容1

证书

常规 详细信息 证书路径

显示(S): <所有>

字段	值
版本	V1
序列号	0dfa
签名算法	sha1RSA
签名哈希算法	sha1
颁发者	support@frank4dd.com, ...
有效期从	2012年8月22日 13:26:54
到	2017年8月21日 13:26:54
使用者	www.example.com, Frank...
公钥	RSA (512 Bits)

编辑属性(E)... 复制到文件(C)...

确定

证书内容2

证书

常规 详细信息 证书路径

显示(S): <所有>

字段	值
签名哈希算法	sha1
颁发者	support@frank4dd.com, ...
有效期从	2012年8月22日 13:26:54
到	2017年8月21日 13:26:54
使用者	www.example.com, Frank...
公钥	RSA (512 Bits)
公钥参数	05 00
指纹	071cb94f0cc8514d024124...

编辑属性(E)... 复制到文件(C)...

确定

解析结果如下：

```
1 版本(version):
2     1
3 序列号(Serial Number):
4     3578(0xdfa)
5 签名算法OID(Algorithm ID):
6     1.2.840.113549.1.1.5
7 颁发者(Issuer):
8     EMAILADDRESS=support@frank4dd.com, CN=Frank4DD web CA, OU=webCert Support,
9     O=Frank4DD, L=Chuo-ku, ST=Tokyo, C=JP
10 有效期从(Not Before):
11     2012/08/22
    到(Not After):
```

```
12      2017/08/21
13 (Subject):
14      CN=www.example.com, O=Frank4DD, ST=Tokyo, C=JP
15 公钥算法(Public Key Algorithm):
16      RSA
17 公钥(Subject Public Key):
18      305c300d06092a864886f70d01010105
19      00034b0030480241009bfc6690798442
20      bbab13fd2b7bf8de1512e5f193e3068a
21      7bb8b1e19e26bb9501bfe730ed648502
22      dd1569a834b006ec3f353c1e1b2b8ffa
23      8f001bdf07c6ac53070203010001
24 签名算法(Certificate Signature Algorithm):
25      SHA1withRSA
26 签名(Certificate Signature Algorithm):
27      14b64cbb817933e671a4da516fcb081d
28      8d60ecbc18c7734759b1f22048bb61fa
29      fc4dad898dd121ebd5d8e5bad6a636fd
30      745083b60fc71ddf7de52e817f45e09f
31      e23e79eed73031c72072d9582e2afe12
32      5a3445a119087c89475f4a95be23214a
33      5372da2a052f2ec970f65bfafddfb431
34      b2c14a9c062543a1e6b41e7f869b1640
35
36 证书扩展域:
37      null
38 证书扩展域2:
39      null
40
```

二、X.509证书结构概述

根据官方文档中给出的结构，X.509使用ASN1描述，其结构描述如下：

```
1 Certificate ::= SEQUENCE {
2     tbsCertificate      TBSCertificate,
3     signatureAlgorithm  AlgorithmIdentifier,
4     signatureValue      BIT STRING }
5
6 TBSCertificate ::= SEQUENCE {
7     version              [0] EXPLICIT Version DEFAULT v1,
8     serialNumber         CertificateSerialNumber,
9     signature            AlgorithmIdentifier,
10    issuer                Name,
11    validity              Validity,
12    subject               Name,
13    subjectPublicKeyInfo  SubjectPublicKeyInfo,
14    issuerUniqueID        [1] IMPLICIT UniqueIdentifier OPTIONAL,
15                          -- If present, version MUST be v2 or v3
16    subjectUniqueID       [2] IMPLICIT UniqueIdentifier OPTIONAL,
17                          -- If present, version MUST be v2 or v3
18    extensions            [3] EXPLICIT Extensions OPTIONAL
19 }
```

```

19          -- If present, version MUST be v3
20      }
21
22  Version ::= INTEGER { v1(0), v2(1), v3(2) }
23
24  CertificateSerialNumber ::= INTEGER
25
26  Validity ::= SEQUENCE {
27      notBefore      Time,
28      notAfter       Time }
29
30  Time ::= CHOICE {
31      utcTime        UTCTime,
32      generalTime    GeneralizedTime }
33
34  UniqueIdentifier ::= BIT STRING
35
36  SubjectPublicKeyInfo ::= SEQUENCE {
37      algorithm       AlgorithmIdentifier,
38      subjectPublicKey BIT STRING }
39
40  Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
41
42  Extension ::= SEQUENCE {
43      extnID          OBJECT IDENTIFIER,
44      critical        BOOLEAN DEFAULT FALSE,
45      extnValue       OCTET STRING
46                      -- contains the DER encoding of an ASN.1 value
47                      -- corresponding to the extension type identified
48                      -- by extnID
49  }

```

ASN1语法

ASN1使用了TLV模式描述，即 `tag`, `length`, `value` 模式。

读取TLV时：

- 首先一个 `byte` 读出 `tag` 值，可根据 `tag` 判断接下来的数据的类型；
- 然后读出 `length` 值(具体读取方法下面说)；
- 然后根据 `length` 的值读取接下来的 `length` 个字节，这是 `value` 字段的长度。

读取到 `value` 字段后，可以进行一下步的解析。

读取 `tag` 时，我们要根据ASN1的语法定义来判断数据类型，根据X690的标准，再结合X.509语法规范，大致的 `tag` 与类型的对照表如下：

类型	tag值	解释
BOOL	0x01	布尔类型，其值(value)为真时 0xff ,为假时 0x00
INTEGER	0x02	整型
BIT_STRING	0x03	比特串，不足字节时要在最后补全为1字节
OCTET_STRING	0x04	字符串
ASN_NULL	0x05	NULL
OBJECT_IDENTIFIER	0x06	对象OID，可以查询OID库
PRINTABLE_STRING	0x13	可打印的字符串，ASCII形式
UTC_TIME	0x17	时间类型
GENERALIZED_TIME	0x18	时间类型
SEQUENCE	0x30	(构造类型) 序列
SET	0x31	(构造类型) 集合

读取 length 时，我们需要分析 length 存入的策略：

- 如果 length ≤ 127 ，那么我们将 length 的值存到1个字节中(低7位)；
- 如果 length > 127 ，那么我们将使用多个字节来存长度值，首先将第一个字节的最高位为1，然后我们使用第一个字节的剩下7位表示接下来有多少个字节用来表示长度；最后我们将长度值存入接下来的几个字节中。

value 中即存储了当前字段的值的信息。

要注意，如果我们读取的是基本类型，那么 value 中就直接存储了对应的值，如果我们读取的是构造类型，那么 value 中往往是一系列的其他的TLV结构，此时需要递归的调用解析。

X.509结构

按序来看，存储的信息大致分别有

1. 版本信息 (Version)
2. 序列号 (SerialNumber)
3. 特征算法的OID (AlgorithmIdentifier)
4. issuer
5. 有效期 (Validity)：内部存储了开始时间和结束时间 (可选 UTCTime 或 GeneralizedTime 类型)
6. 证书主体信息 (subject)
7. 公钥信息 (SubjectPublicKeyInfo)
8. 扩展部分
9. 特征算法 (又一次存储了特征算法)
10. 特征值 (可根据特征算法来解析， BITSTRING 类型)

使用一个 testCA 版本的证书转为16进制查看，我们可以尝试着自己读取其信息：

其部分信息如下（学习时自己标注，后来明白了结构就没有标注后边的内容）：

```
1  30 82 03 EE { SQ 03EE长 } ==> (Certificate)
2    30 82 02 D6 { SQ 02D6长 } ==> (TBSCertificate)
3      A0 03 { VERSION 03长 } ==> (Version v3=0x02)
4        02 01 02 { 整数 01长 值02 }
5          02 10 {整数 10长 } ==> (CertificateSerialNumber{int})
6            2B 85 F2 FE 98 D1 76 99 4F 38 BF AB 9D A6 2D 5F {值 2B 85 ...}
7          30 0D { SQ 0D 长} ==> (AlgorithmIdentifier)
8            06 09 { OID 09长 } ==> (algorithm{OID})
9              2A 86 48 86 F7 0D 01 01 05 {值}
10             05 00 { NULL 00长 } ==> (parameters{optional})
11          30 55 { SQ 55长 } ==> (issuer{Name})
12            31 0B {SET 0B长 }
13              30 09 {SQ 09长 }
14                06 03 {OID 03长} ==>
15                  55 04 06 {值}
16                    13 02 43 4E
17              31 0B {SET 0B长 }
18                30 09 06 03 55 04 08 13 02 53 43
19              31 0B 30 09 06 03
20                55 04 07 13 02 43 44 31 0E 30 0C 06 03 55 04 0A
21                13 05 55 45 53 54 43 31 0B 30 09 06 03 55 04 0B
22                13 02 43 53 31 0F 30 0D 06 03 55 04 03 13 06 74
23                65 73 74 43 41
24          30 1E {SQ 1E长} ==> (Validty)
25            17 0D { UTCTIME 0D长 } ==> (notBefore)
26              31 35 30 35 32 33 30 33 34 33 33 31 5A {值}
27            17 0D { UTCTIME 0D长 } ==> (notAfter)
28              32 30 30 35 32 33 30 33 35 32 31 34 5A {值}
29          30 55 {SQ 55长} ==> (Subject)
30            31 0B {SET 0B长}
31              30 09 {SQ 09长}
32                06 03 55 04 06 13 02 43 4E
33              31 0B {SET 0B长}
34                30 09 {SQ 09长}
35                  06 03 55 04 08 13 02 53 43
36              31 0B {SET 0B长}
37                30 09 {SQ 09长}
38                  06 03 55 04 07 13 02 43 44
39 .....
40 .....
41 .....
42
```

在上述中我们可以看出来证书的结构。

解析程序

我们使用 java 的 `x509Certificate` 和 `CertificateFactory` 两个库进行解析。

提供了可执行程序 `jar` 包供使用，请使用命令行参数指定被解析的文件。

使用示例如下：

