X.509_Reader程序设计与实现

数据科学与计算机学院 17343094 彭湃

上次的信息安全作业，我们实现了`MD5`算法加密程序的设计与实现，本次作业，我们需要设计实现一个小程序，对`X.509`证书进行读写，并且输出其中的相关信息，下面开始我们的正式设计实现。

一、X.509证书结构分析

在密码学中，`X.509`是定义公用秘钥证书格式的标准。`X.509`证书用于许多Internet协议，包括`TLS`和`SSL`，可以说它是`HTTPS`的基础。

> `X.509`证书里含有公钥、身份信息（比如网络主机名，组织的名称或个体名称等）和签名信息（可以是证书签发机构CA的签名，也可以是自签名）。对于一份经由可信的证书签发机构签名或者可以通过其它方式验证的证书，证书的拥有者就可以用证书及相应的私钥来创建安全的通信，对文档进行数字签名。
>
> --摘自`WIKI`百科

`X.509`证书是用`ASN.1`定义的，同时它的组成结构标准也可以用`ASN.1`来进行描述，具体的定义如下：

```
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,
                     -- If present, version must be v2 or v3
subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                     -- If present, version must be v2 or v3
```

```
extensions      [3]  EXPLICIT Extensions OPTIONAL
                     -- If present, version must be v3
}
```

目前的证书已经基本是`V3`版本，下面我们将会对其的具体结构进行分析：

- 版本号(`version`)：有`V1`、`V2`、`V3`三个版本，目前我看到的基本都是`V3`版本。
- 序列号(`serialNumber`)：可以理解为是证书颁发机构给证书的标识符。
- 签名(`signature`)：证书的数字签名。
- 颁发机构(`issuer`)：就是`CA`的名称。
- 使用者(`subject`)：证书使用机构的名称及信息。
- 颁发者唯一标识(`issuerUniqueID`)：作为`CA`的唯一标识符，如果没有的话就返回`null`。
- 使用者唯一标识(`subjectUniqueID`)：作为使用者的唯一标识符，如果没有则返回`null`。
- 证书有效期(`validity`)：检验证书是否有效，即证书是否还在有效期内，通过起始时间和截止时间来确认。
    - 此日期前(`NotBefore`)无效
    - 此日期后(`NotAfter`)无效
- 签名算法(`AlgorithmID`)：由对象标识符和相关参数组成。
- 主体公钥信息(`subjectPublicKeyInfo`)
    - 公钥算法
    - 主体公钥
- 拓展信息(`extensions`)：即为这个证书的拓展信息（不必需）

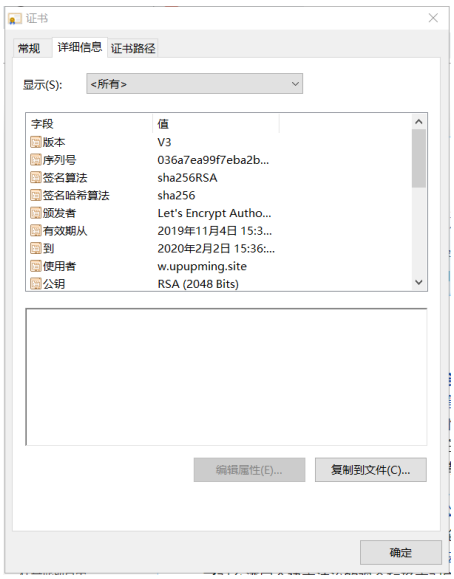
基本就这些了，构成一个`X.509`的证书应该够了哈哈哈

二、获取X.509证书

为了后期读取程序效果的验证，我们首先需要获取一批证书文件，之前以为步骤会很复杂，后来查找资料发现是真的挺容易的，其实只有几步：

- 用`chrome`打开我们想要获取证书的网站。
- 点击网址导航栏前面的小锁状的安全符号，可以看到详细的安全信息。

- 点击**证书(有效)**，然后点击**详细信息**，可以看到下面的页面。



- 点击复制到文件，选择`DER 编码二进制 X.509`，选择一个文件复制下来。

- 至此我们就可以获得这个网站的`X.509`证书的`.cer`文件。

三、程序数据结构

在查找`X.509`相关信息的时候，意外的发现了`java`简直是读取证书的神器，自带`certificate`、`X509Certificate`、`CertificateFactory`等类，相比较，用`C/C++`则麻烦的多，无论是信息的读取还是后期数据的处理，因为本次作业我们只要求实现一个读取程序，而不限定使用什么语言，所以我们就直接使用`java`开发好吧。

下面我们将分部分对我们的程序进行阐述分析：

1. 相关类的介绍

- `java.security.cert`：这个包主要提供用于解析和管理证书、证书撤消列表(`CRL`)和证书路径的类和接口。它包含对`X.509 v3`证书和`X.509 v2 CRL`的支持。这个包的功能非常强大，我们本次的任务也恰恰好正是对`X.509 V3`证书进行读取解析，所以这个包是我们绕不开的一个话题。

  关于这个包更加详细的介绍可以直接在`java`的官方文档中看到(做一回搬运工)：官方链接

  而我们使用的主要是下面的几个功能强大的类：

  - `Certificate`:一种抽象类，用于管理各种身份证书(我们所说的`X.509`也是其中一种)，不同的证书，例如`X.509`、`PGP`、`SDSI`等都可以通过对这个类进行子类化来实现具体的功能。目前最知名的子类应该就是`X509Certificate`类，我们下个小部分会重点介绍它。

比较常用的功能函数是：

函数	功能
public int hashCode()	从证书的编码形式返回此证书的哈希码值
public abstract byte[] getEncoded()	返回给定证书的编码形式，例如X.509会返回ASN.1形式的
public abstract String toString()	这个用的真的比较多，返回字符串的表现形式
getPublicKey	从证书中获取它的公钥

事实上……我们最后用到的只有`getEncoded`、`getPublicKey`、`toString`三个函数。在使用相关函数前，需要引入这个类：

```
import java.security.cert.Certificate;
```

- `X509Certificate`：这个就是我们上面说的`Certificate`最著名的子类，它给我们提供了访问`X.509`所有属性的标准方法。下面的很多函数在读取`X.509`证书的不同信息时非常方便，下面我将他们简单列举出来：

函数	功能
public abstract void checkValidity()	主要用来检查证书当前是否还在有效期
public abstract int getVersion()	从`X.509`证书中获取版本号，返回值为`int`
public abstract BigInteger getSerialNumber()	从`X.509`证书中获取序列号，返回值是大整数
public abstract Principal getIssuerDN()	也可以用`getIssuerX500Principal`，将颁发机构作为特定的对象返回
public X500Principal getIssuerX500Principal()	返回证书中颁发机构的值，从1.4开始可以使用这个办法
public abstract Principal getSubjectDN()	也可以用`getSubjectX500Principal`，将使用机构的信息作为特定的对象返回
public X500Principal getSubjectX500Principal()	返回证书中使用机构的值，从1.4开始可以使用这个方法
public abstract Date getNotBefore()	从证书的有效期获取`notBefore`日期，即有效期的起始时间

- 接上，因为函数比较多，为了排版我们划分成两个表格，剩下的部分在下面。

函数	功能

函数	功能
public abstract Date getNotAfter()	从证书的有效期获取`notAfter`日期，即有效期的截止时间
public abstract byte[] getSignature()	从证书中获取该证书的签名值
public abstract String getSigAlgName()	获取证书签名算法的签名算法名称，例如最常用的`SHA256withRSA`
public abstract boolean[] getIssuerUniqueID()	返回证书中的颁发机构的唯一标识符，如果没有的话，我们就直接返回`null`值
public abstract boolean[] getSubjectUniqueID()	返回证书中使用机构的唯一标识符，如果没有的话，我们就直接返回`null`值

- `CertificateFactory`：这个类定义了证书工厂的相关功能，证书工厂主要用来从编码中生成证书、证书路径以及证书吊销列表(`CRL`)。`X.509`的证书工厂必须返回由上面我们说的`X509Certificate`实例的证书以及`X509CRL`实例的`CRL`。

  下面给出一个比较简单的解析证书文件，并且从中提取所有证书的例子：

  ```
  FileInputStream fis = new FileInputStream(filename);
  CertificateFactory cf = CertificateFactory.getInstance("X.509");
  Collection c = cf.generateCertificates(fis);
  Iterator i = c.iterator();
  while (i.hasNext()) {
     Certificate cert = (Certificate)i.next();
     System.out.println(cert);
  }
  ```

2. 模块分解介绍

A. 导入相关类

这个部分没有什么好说的，直接附上代码就好

```
import java.util.Scanner;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.PublicKey;
import java.security.cert.X509Certificate;
import java.security.cert.Certificate;
import java.security.cert.CertificateFactory;
import java.security.cert.CertificateException;
```

B. 交互部分

为了方便解析不同的证书来验证我们的功能，我们设定了通过输入`Y`和`N`来完成不同的简易功能，如果输入`Y`，则可以输入`certificate`地址进行相应证书的解析，如果输入`N`，则直接`exit(0)`退出当前程序：

```
while(true)
{
    Scanner choose = new Scanner(System.in);
    System.out.println("\n\n是否开始解析证书：Y(开始解析)/N(退出程序)");
    String peng = choose.nextLine();
    char ch = peng.charAt(0);

    switch(ch){
        case 'Y' :
            Scanner input = new Scanner(System.in);
            System.out.println("\n请输入你想解析的证书地址：");
            String str = input.next();
            ..... /*具体实现解析证书的代码*/
            break;
         case 'N' :
            System.exit(0);
            break;
    }
}
```

C.实例化证书

证书必须使用证书工厂进行实例化，事实上，怎么进行实例化，官网上给出了很详细的介绍，我们照猫画虎就好。

```
FileInputStream fs = new FileInputStream(str);
CertificateFactory cf = CertificateFactory.getInstance("X.509");
Certificate cer = cf.generateCertificate(fs);
X509Certificate pai = (X509Certificate)cer ;
```

D. 证书内容解析

使用`X509Certificate`类中的函数进行对应的解析即可，具体的解析代码是：

```
// 检查合法性
pai.checkValidity();
// 获得版本
System.out.println("版本:            " + pai.getVersion());
// 获得序列号
System.out.println("序列号:          " + pai.getSerialNumber());
//签名使用算法
System.out.println("签名使用算法:    "+ pai.getSigAlgName());
//证书颁发者的值获取
System.out.println("颁发者:          " + pai.getIssuerDN().toString());
System.out.println("颁发者唯一标识: " + pai.getIssuerUniqueID());
//获取证书有效期
System.out.println("证书有效期从:    "+ pai.getNotBefore());
System.out.println("到:              "+ pai.getNotAfter());
```

```
//证书使用者的值获取
System.out.println("使用者:          " + pai.getSubjectDN());
System.out.println("使用者唯一标识: " + pai.getSubjectUniqueID());
//主题的签名
System.out.println("签名:            "+ pai.getSignature().toString());
// 从此证书中获得公钥
byte [] encode = pai.getPublicKey().getEncoded();
System.out.println("公钥:          ");
String key = "[";
for(int i = 0 ; i < encode.length ; i++)
{
    int index = encode[i] & 0xff;
    key += Integer.toString(index);
    if(i < encode.length - 1) key += ", ";
    if(i % 40 == 0 && i != 0){
        key += "\n";
    }
}
System.out.println(key + "]\n");
```

E.异常处理

由于我们使用了`factory`类型的数据，所以我们需要设置异常抛出，主要是为了防止证书过期或者I/O操作有问题等.....

```
public static void main(String args[]) throws IOException
```

```
catch (CertificateException e) {
    e.printStackTrace();
}
```

四、程序源代码

见**/src/X509.java** 其实上面的模块分解板块已经基本列举出来了，在这里就不过分赘述。

五、编译运行

运行指令为：

```
cd Desktop/X509/src
javac X509.java
java X509
```

在上面的部分，我们已经介绍过怎样获取证书，下面我将会用我日常接触最多的三个网站证书作为实验来读取解析，分别是`github`、`bilibili`、`网易云(NeteaseMusic)`。他们的`cer`文件可以直接在**/cer**文件夹下面找到。

- 读取解析`github.cer`：

```
Y
../cer/github.cer
```

下面是解析后的结果：

```
是否开始解析证书：Y(开始解析)/N(退出程序)
Y

请输入你想解析的证书地址：
./cer/github.cer
版本:         3
序列号:       133244125631355695976993629735395177 27
签名使用算法:   SHA256withRSA
颁发者:        CN=DigiCert SHA2 Extended Validation Server CA, OU=www.digicert.com, O=DigiCert Inc, C=US
颁发者唯一标识: null
证书有效期从:   Tue May 08 08:00:00 CST 2018
到:           Wed Jun 03 20:00:00 CST 2020
使用者：        CN=github.com, O="GitHub, Inc.", L=San Francisco, ST=California, C=US, SERIALNUMBER=5157550,
ganization
使用者唯一标识：null
签名:          [B@28b689e0
公钥:
[48, 130, 1, 34, 48, 13, 6, 9, 42, 134, 72, 134, 247, 13, 1, 1, 1, 5, 0, 3, 130, 1, 15, 0, 48, 130, 1, 10, 2,
193, 79, 40, 173, 114, 112, 125, 211, 206, 185, 181, 96, 115, 164, 116, 155, 138, 119, 70, 253, 122, 152, 66,
195, 147, 243, 249, 117, 144, 188, 191, 187, 224, 149, 186, 46, 197, 141, 115, 97, 5, 211, 16, 132, 168, 179,
253, 119, 97, 218, 155, 27, 154, 35, 255, 140, 126, 162, 1, 6, 221, 209, 127, 83, 150, 8, 193, 90, 250, 231,
172, 182, 74, 156, 193, 234, 232, 251, 150, 64, 41, 246, 21, 48, 181, 4, 176, 204, 5, 182, 132, 195, 36, 89,
213, 204, 170, 58, 133, 5, 82, 6, 50, 150, 7, 97, 223, 39, 130, 12, 247, 133, 219, 96, 49, 240, 9, 80, 197, 1
51, 4, 246, 81, 63, 82, 152, 21, 233, 11, 118, 71, 92, 77, 74, 107, 197, 8, 21, 174, 248, 209, 87, 233, 234,
140, 128, 187, 112, 86, 145, 15, 75, 2, 3, 1, 0, 1]
```

- 读取解析`bilibili.cer`：

```
Y
../cer/bilibili.cer
```

```
是否开始解析证书：Y(开始解析)/N(退出程序)
Y

请输入你想解析的证书地址：
./cer/bilibili.cer
版本:         3
序列号:       122028429346224628963170105 20
签名使用算法:   SHA256withRSA
颁发者:        CN=GlobalSign Organization Validation CA - SHA256 - G2, O=GlobalSign nv-sa, C=BE
颁发者唯一标识: null
证书有效期从:   Tue Sep 18 17:32:07 CST 2018
到:           Fri Sep 18 17:21:04 CST 2020
使用者：        CN=*.bilibili.com, O=上海幻电信息科技有限公司, L=上海, ST=上海, C=CN
使用者唯一标识：null
签名:          [B@75af8109
公钥:
[48, 130, 1, 34, 48, 13, 6, 9, 42, 134, 72, 134, 247, 13, 1, 1, 1, 5, 0, 3, 130, 1, 15, 0, 48, 13
40, 193, 76, 142, 21, 172, 34, 90, 138, 49, 128, 15, 32, 59, 29, 169, 166, 210, 118, 113, 37, 160
242, 240, 91, 196, 105, 202, 111, 170, 213, 235, 134, 167, 6, 47, 103, 43, 147, 210, 112, 51, 69,
131, 254, 179, 190, 105, 32, 155, 32, 93, 221, 26, 2, 13, 83, 232, 42, 145, 122, 132, 197, 18, 10
97, 142, 165, 135, 36, 124, 50, 89, 53, 13, 44, 46, 128, 109, 241, 164, 150, 29, 18, 170, 201, 16
35, 135, 139, 159, 107, 65, 210, 82, 172, 24, 101, 216, 111, 217, 160, 67, 230, 233, 69, 162, 129
12, 67, 84, 23, 233, 121, 125, 61, 211, 107, 191, 43, 210, 2, 138, 147, 124, 19, 143, 31, 79, 98,
231, 7, 29, 170, 53, 90, 200, 249, 2, 3, 1, 0, 1]
```

- 读取解析`NeteaseMusic.cer`：

```
Y
../cer/NeteaseMusic.cer
```

```
是否开始解析证书：Y(开始解析)/N(退出程序)
Y

请输入你想解析的证书地址：
./cer/NeteaseMusic.cer
版本:             3
序列号:           432573630281317443316880562822662079 5
签名使用算法:     SHA256withRSA
颁发者:           CN=GeoTrust RSA CA 2018, OU=www.digicert.com, O=DigiCert Inc, C=US
颁发者唯一标识: null
证书有效期从:     Thu Dec 20 08:00:00 CST 2018
到:               Fri Mar 20 20:00:00 CST 2020
使用者：          CN=*.163.com, OU=Game Dep., O="NetEase (Hangzhou) Network Co.,Ltd", L=Hangzhou, ST=Zhejiang, C=CN
使用者唯一标识：null
签名:             [B@a756b37
公钥:
[48, 130, 1, 34, 48, 13, 6, 9, 42, 134, 72, 134, 247, 13, 1, 1, 1, 5, 0, 3, 130, 1, 15, 0, 48, 130, 1, 10, 2, 130,
160, 223, 149, 145, 215, 13, 14, 82, 134, 191, 66, 185, 75, 107, 41, 235, 55, 143, 38, 16, 255, 49, 15, 224, 100,
37, 191, 87, 78, 251, 52, 167, 241, 208, 35, 172, 22, 18, 245, 134, 6, 255, 241, 180, 11, 229, 226, 251, 194, 31,
20, 31, 140, 27, 61, 115, 97, 67, 21, 79, 61, 21, 235, 229, 190, 152, 166, 15, 179, 70, 140, 201, 152, 123, 32, 23
139, 200, 24, 238, 242, 132, 186, 125, 173, 147, 245, 116, 120, 192, 95, 66, 248, 49, 195, 109, 100, 125, 78, 128,
190, 50, 245, 140, 136, 99, 107, 229, 3, 28, 237, 246, 186, 245, 159, 222, 233, 209, 253, 22, 204, 35, 147, 134, 1
13, 40, 171, 201, 43, 199, 1, 182, 160, 209, 185, 188, 29, 170, 19, 41, 114, 25, 203, 119, 12, 119, 36, 132, 254,
232, 173, 97, 144, 182, 231, 85, 143, 2, 3, 1, 0, 1]
```

至此我们的整个实验结束。