X.509证书的读取操作与分析

1. X.509

该部分所使用知识均来自于维基百科与 CSDN

1.1 定义

X.509 是密码学里公钥证书的格式标准。

X.509 证书已应用在包括 `TLS/SSL` 在内的众多 Intenet 协议里.同时它也用在很多非在线应用场景里，比如电子签名服务。

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.

另外除了证书本身功能，X.509还附带了证书吊销列表和用于从最终对证书进行签名的证书签发机构直到最终可信点为止的证书合法性验证算法。

1.2 组成结构

- 证书
  - 版本号
    - 作用：【标识证书的版本（版本1、版本2、或是版本3）】
  - 序列号
    - 作用：【标识证书的唯一整数，由证书颁发者分配的本证书的唯一标识符】
  - 签名算法
    - 作用：【由于签名书的算法标识，由对象标识符加上相关的参数组成，用于说明本证书所用的数字签名算法。例如，SHA-1 和 RSA 的对象标识符就用来说明该数字签名是利用 RSA 对SHA-1 杂凑加密】
  - 颁发者
    - 作用：【证书颁发者的可识别名】
  - 证书有效期
    - 作用：【证书的有效期时间段】
      - “Not Before” 此日期前无效
      - “Not After” 此日期后有效
      - 以上二者分别由 `UTC` 时间或一般的时间表示
  - 主体
    - 作用：【证书拥有者的可识别名，这个字段必须是非空的，除非你在证书扩展中有别名】
  - 主体公钥信息
    - 作用：【标识主题的公钥以及算法标识符】
      - 公钥算法

        - 主题公钥
    - 颁发者唯一身份信息（可选项）
        - 作用：证书颁发者的唯一标识符，仅在版本 2 与版本 3 中有要求，属于可选项
    - 主题唯一身份信息
        - 作用：证书拥有者的唯一标识符，仅在版本 2 和版本 3 中有要求，属于可选项
    - 扩展信息 （**次重点部分**）
        - 发行者秘钥标识符
            - 作用：【证书所含密钥的唯一标识符，用来区分同一证书拥有者的多对密钥】
        - 秘钥使用
            - 作用：【一个比特串，指明（限定）证书的公钥可以完成的功能或服务，如：证书签名、数据加密等。

              如果某一证书将 KeyUsage 扩展标记为“极重要”，而且设置为“keyCertSign”，则在 SSL 通信期间该证书出现时将被拒绝，因为该证书扩展表示相关私钥应只用于签写证书，而不应该用于 SSL。】
        - CRL 分布点
            - 作用：【指明 CRL 的分布地点】
        - 私钥的使用期
            - 作用：【指明证书中与公钥相联系的私钥的使用期限，它也由 “ Not Before “ 和" Not After "组成。若此项不存在时，公私钥的使用期是一样的】
        - 证书策略
            - 作用：【由对象标识符和限定符组成，这些对象标识符说明证书的颁发和使用策略有关】
        - 策略映射
            - 作用：【表明两个 CA 域之间的一个或多个策略对象标识符的等价关系，仅在 CA 证书里存在】
        - 主体别名
            - 作用：【指出证书拥有者的别名，如电子邮件地址、IP 地址等，别名是和 DN 绑定在一起的】
        - 颁发者别名
            - 作用：【指出证书颁发者的别名，如电子邮件地址、IP地址等，但颁发者的 DN 必须出现在证书的颁发者字段】
        - 主体目录属性
            - 作用：【指出证书拥有者的一系列属性。可以使用这一项来传递访问控制信息】
- 证书签名算法
- 数字签名

1.3 安全性

- 采用黑名单方式的证书吊销列表 CRL 和在线证书状态协议(OCSP)
    - 如果客户端仅信任在CRL可用的时候信任证书，那就失去离线信任的需求。因此通常客户端会在CRL不可用的情况下信任证书，因而给了那些可以控制信道的攻击者可乘之机。如谷歌的Adam Langley所说，对CRL的检查有如你期望安全带在出事故事一定能正常使用的
- 在大范围及复杂的分布模式下选用CRL并不明智
- OCSP由于没有吊销状态的历史记录也会出现歧义

- **聚合问题**
- **代表问题**: 证书颁发机构事没办法限制其下属颁发的证书作出名字及属性方面的限制。而且在Internet上存在着相当多的证书颁发机构，想对他们进行分类和策略上的限制是一项不可能完成的任务。
- **分布问题**: 证书链引的下属颁发机构，桥接颁发机构以及交叉认证使得证书验证变得非常复杂，需要付出很大的代价。层次式的第三方信任模型作为一种唯一的模型的话，路径验证也可能出现含糊不明的情况岐义，这对于已经创建双边信任也很不方便。
- 发布一个对主机名的扩展验证并不能防止再发布一个验证要求低一些的适用于同一个主机名的证书。这就造成了不能对中间人攻击的有效保护

1.4 证书文件名扩展类型

X.509有多种常用的扩展名。不过其中的一些还用于其它用途，就是说具有这个扩展名的文件可能并不是证书，比如说可能只是保存了私钥。

- `.pem` – (**隐私增强型电子邮件**) **DER**编码的证书再进行 **Base64** 编码的数据存放在"-----BEGIN CERTIFICATE-----"和"-----END CERTIFICATE-----"之中
- `.cer, .crt, .der` – 通常是**DER**二进制格式的，但 **Base64** 编码后也很常见。
- `.p7b, .p7c` – **PKCS#7**
  - 注：`PKCS#7` 是签名或加密数据的格式标准，官方称之为容器。由于证书是可验真的签名数据，所以可以用 **SignedData** 结构表述。
  - 注：`P7C` 文件是退化的 **SignedData** 结构，没有包括签名的数据。
- `.p12` – **PKCS#12**格式，包含证书的同时可能还有带密码保护的私钥
  - 注：`PKCS#12` 由 **PFX** 进化而来的用于交换公共的和私有的对象的标准格式。
- `.pfx` – **PFX，PKCS#12**之前的格式（通常用 **PKCS#12** 格式，比如那些由 **IIS** 产生的 **PFX** 文件）

---

2. 读取操作程序

2.1 程序语言选择

语言：`Java`

选择原因：

- 一开始打算用 `C++` 实现，使用 `C++` 内置的二进制读写函数完成对文件证书的读取，随后再写一些函数进行操作，但是后来发现读取时候， `C++` 库文件并不支持读取 **.cer** 类型的文件读取，采用断点调试才发现问题，程序进入这一步，直接就运行错误强行中断；搜索了一些资料之后，发现要是想使用 `C++` 进行操作，还得需要做很多其他的操作，过程太过于繁琐，所以就放弃了继续使用 `C++`
- 随后向已经实习的师兄求助，他让我去搜索了 `Java` 的 `CertificateFactory` 类方法，然后我自己去简单查阅了一下文件，发现确实是 `Java` 中有现成的对 `X.509证书` 进行操作的很多函数，包括读写以及对某一项数据的输出等，所以最后就选择了使用 `Java` 语言；并且在读入 `X.509证书` 之后不会再出现乱码跟程序崩溃的情况了；
- 在撰写这次实验报告时候，也是把自己在学习 `X509` 时候的资料都放了进来，进行了一次系统整理，方便自己在期末进行复习

2.2 编译环境

操作系统： windows 10

条件：

- 安装最新版 jre、jdk
- 配置环境变量
- 编译 javac X509.java
- 运行 java X509

2.3 Security 方法

参考链接

2.4 Security.cert 方法

参考链接

2.5 Security.cert.CrtificateFactory 方法

`public class CertificateFactory extends Object`

此类定义了用于从相关的编码中生成证书、证书路径 (`CertPath`) 和证书撤消列表 (CRL) 对象的 `CertificateFactory` 功能。

为了实现多个证书组成的编码，如果要解析一个可能由多个不相关证书组成的集合时，应使用 `generateCertificates` 。否则，如果要生成 `CertPath` （证书链）并随后使用 `CertPathValidator` 验证它，则应使用 `generateCertPath` 。

(X.509 的 `CertificateFactory` 返回的证书必须是 `java.security.cert.X509Certificate` 的实例)

以下示例代码解析一个存储在文件中的 PKCS#7 格式的证书答复，并从中提取所有的证书：

```
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);
 }
```

2.6 获取对应数据

在 2.5 中已经成功的创建一个 `X509Certificate` 类型的对象，接着我们要做的就是读取到对应的数据，所用的函数方法部分如下表所示：

方法摘要	
abstract void	checkValidity() 检查证书目前是否有效。
abstract void	checkValidity(Date date) 检查给定的日期是否处于证书的有效期内。
abstract int	getBasicConstraints() 从关键 BasicConstraints 扩展 (OID = 2.5.29.19) 中获得证书的限制路径长度。
List<String>	getExtendedKeyUsage() 获得一个不可修改的 String 列表，表示已扩展的密钥使用扩展 (OID = 2.5.29.37) 中 ExtKeyUsageSyntax 字段的对象标识符（OBJECT IDENTIFIER）。
Collection<List<?>>	getIssuerAlternativeNames() 从 IssuerAltName 扩展 (OID = 2.5.29.18) 中获得一个发布方替换名称的不可变集合。
abstract Principal	getIssuerDN() **已过时**，由 getIssuerX500Principal() 替代。
abstract boolean[]	getIssuerUniqueID() 获得证书的 issuerUniqueID 值。
X500Principal	getIssuerX500Principal() 以 X500Principal 的形式返回证书的发布方（发布方标识名）值。
abstract boolean[]	getKeyUsage() 获得一个表示 KeyUsage 扩展 (OID = 2.5.29.15) 的各个位的 boolean 数组。
abstract Date	getNotAfter() 获得证书有效期的 notAfter 日期。
abstract Date	getNotBefore() 获得证书有效期的 notBefore 日期。
abstract BigInteger	getSerialNumber() 获得证书的 serialNumber 值。
abstract String	getSigAlgName() 获得证书签名算法的签名算法名。

```
System.out.println("输出证书信息:\n" + s ) ;
System.out.println("版本号:" + t.getVersion()) ;
System.out.println("序列号:" + t.getSerialNumber().toString(16)) ;
System.out.println("签发者:"+ t.getIssuerDN()) ;
System.out.println("有效起始日期:"+ t.getNotBefore()) ;
System.out.println("有效终止日期:"+ t.getNotAfter()) ;
System.out.println("主体名:"+t.getSubjectDN()) ;
System.out.println("签名算法:"+t.getSigAlgName()) ;
System.out.println("签名:"+t.getSignature().toString()) ;
.....
```

3. 运行结果

3.1 编译时遇到的问题与解决方案

3.1.1 字符编码问题

由于在`println`使用了部分中文字符输出，所以在第一次进行编译时候出现如下错误：

```
PS C:\Users\WYX\Desktop\X509\代码> javac .\X509.java
.\X509.java:1: 错误: 编码GBK的不可映射字符
//濮煎瓑鎵?闀?瑕佺殑鍖?

.\X509.java:1: 错误: 编码GBK的不可映射字符
//濮煎瓑鎵?闀?瑕佺殑鍖?

.\X509.java:1: 错误: 编码GBK的不可映射字符
//濮煎瓑鎵?闀?瑕佺殑鍖?

.\X509.java:8: 错误: 编码GBK的不可映射字符
        //X.509 鐨? CertificateFactory 杩斿洖鐨勮瘉涔﹀繀椤绘槸 java.security.cert.X509Certificate 鐨勫疄渚?

.\X509.java:8: 错误: 编码GBK的不可映射字符
        //X.509 鐨? CertificateFactory 杩斿洖鐨勮瘉涔﹀繀椤绘槸 java.security.cert.X509Certificate 鐨勫疄渚?

.\X509.java:9: 错误: 编码GBK的不可映射字符
        //鐩存帴鏋勫缓Factory绫诲瀷鐨勫  璞?

.\X509.java:11: 错误: 编码GBK的不可映射字符
        // 杩欓噷鎴戜滑浣跨敤鍙傝€冩枃鐚  腑鐨勪唬鐮佹牱渚嬭繘琛屾搷浣?

.\X509.java:11: 错误: 编码GBK的不可映射字符
        // 杩欓噷鎴戜滑浣跨敤鍙傝€冩枃鐚  腑鐨勪唬鐮佹牱渚嬭繘琛屾搷浣?

.\X509.java:24: 错误: 编码GBK的不可映射字符
        //瀹炰緥鍖栨被鍨?

.\X509.java:28: 错误: 编码GBK的不可映射字符
        //杩欓噷?瑕佷娇鐢ㄤ箣鍓嶉渶瑕佸垱寤?)鐨factory绫诲瀷杞  负X509certificate绫诲瀷锛屽洜涓哄悗闈㈤渶瑕佺敤鍒扮瓟杞  喅

.\X509.java:34: 错误: 编码GBK的不可映射字符
        System.out.println("鐗堟湰?: " + t.getVersion()) ;
```

解决方案：

编译时候使用：`javac -encoding UTF-8 .\X509.java`指令将字符进行转

3.1.2 异常处理抛出问题

由于使用了 `factory` 类型的数据，我们要防止证书过期同时也要保证 IO 操作的正确性，所以需要加上错误捕捉，让程序可以在异常出现时直接退出；

错误如图：

```
PS C:\Users\WYX\Desktop\X509\代码> javac -encoding UTF-8 .\X509.java
.\X509.java:23: 错误: 未报告的异常错误FileNotFoundException; 必须对其进行捕获或声明以便抛出
        FileInputStream fis = new FileInputStream("out.cer") ;

.\X509.java:25: 错误: 未报告的异常错误CertificateException; 必须对其进行捕获或声明以便抛出
        factory = CertificateFactory.getInstance("X.509") ;

.\X509.java:27: 错误: 未报告的异常错误CertificateException; 必须对其进行捕获或声明以便抛出
        Certificate c = factory.generateCertificate(fis);

.\X509.java:31: 错误: 未报告的异常错误IOException; 必须对其进行捕获或声明以便抛出
        fis.close();

4 个错误
```

解决方案：

在 main 函数加上异常处理：

```java
public class X509 {
    public static void main(String args[]) throws IOException {
        //X.509 的 CertificateFactory 返回的证书必须是 java.security.cert.X509Certificate 的实例
        //直接构建Factory类型的对象
        CertificateFactory factory ;
        // 这里我们使用参考文献中的代码样例进行操作
```

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

3.2 程序运行

3.2.1 随机证书生成

参考链接

在上述链接中生成的`X509`证书如下：

```
-----BEGIN CERTIFICATE-----
MIIDPTCCAiegAwIBAgIBATALBgkqhkiG9w0BAQUwHjEcMAkGA1UEBhMCUlUwDwYD
VQQDHggAVABlAHMAdDAeFw0xNjAxMzExNjAwMDBaFw0xOTAxMzExNjAwMDBaMB4x
HDAJBgNVBAYTAlJVMA8GA1UEAx4IAFQAZQBzAHQwggEiMA0GCSqGSIb3DQEBAQUA
A4IBDwAwggEKAoIBAQDqbF/+iB4WbqzNs5+VeI38O8JIqpQVlSlTBdRyH3KYdPN3
o54DFOgJIYYOJS5bGisJ5bP1sZQpuGsr/zdYhw9tQOJZvX+3lm5r9MkFNgg+JyjO
4J5+8UrAzxUDLs1suXiogADllYgopLMuuVjY2gNa78OV+OORfOqx5F9INpDDlUh7
LQVqhf3f+zIvjpf8Ast3wTeUm2PrlaO3QnUFK+PRO/8jFj+7le0o89I6JD7Hkw/9
uUG0cfMBC8z3nEJxVjOOYKxUG/di0th8eHnhY2dnm1YxYqNHaHVHUzO2nYeTm1rE
0uFPrb9n7Rx6DVLWD8Xe6oMgGBrBKYozOzzmqYZPAgMBAAGjgYkwgYYwEgYDVR0T
AQH/BAgwBgEB/wIBAzALBgNVHQ8EBAMCAAYwYwYDVR0lBFwwWgYEVR0lAAYIKwYB
BQUHAwEGCCsGAQUFBwMCBggrBgEFBQcDAwYIKwYBBQUHAwQGCCsGAQUFBwMIBggr
BgEFBQcDCQYKKwYBBAGCNwoDAQYKKwYBBAGCNwoDBDALBgkqhkiG9w0BAQUDggEB
AD9q49T5BEvuerU5O1jRlIijsdTn/Z7VROlc5RO4s2rTVv/whtzSuCB+VF81E2YM
sVebLEQ4zGbvPv6Wp4PwWpLxDjnMZ6cVTLa3ZnQgBc7wZJvgID93j2BkcRtwu6f2
SIdeMlp6ZN1F9iRrgF7E1LvyUIqP0kr3JjjItj+J1sszFLub/DCQsmUAhTfzo4wj
z5rDaC8qaLSWvd3nFqX/7WFxGiyiIR+jDfjBi5zKk+W7GsUN4Y1oc5RT96UgicmB
uAsimukOdKVjXpfiRKNRm81OyXVNhNjLpFd+92kwRXH/VuroUDnF87dBTxChbAcq
EPTsdW6lxTMQOg2nVOJfKnI=
-----END CERTIFICATE-----

-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDqbF/+iB4WbqzN
s5+VeI38O8JIqpQVlSlTBdRyH3KYdPN3o54DFOgJIYYOJS5bGisJ5bP1sZQpuGsr
/zdYhw9tQOJZvX+3lm5r9MkFNgg+JyjO4J5+8UrAzxUDLs1suXiogADllYgopLMu
uVjY2gNa78OV+OORfOqx5F9INpDDlUh7LQVqhf3f+zIvjpf8Ast3wTeUm2PrlaO3
QnUFK+PRO/8jFj+7le0o89I6JD7Hkw/9uUG0cfMBC8z3nEJxVjOOYKxUG/di0th8
eHnhY2dnm1YxYqNHaHVHUzO2nYeTm1rE0uFPrb9n7Rx6DVLWD8Xe6oMgGBrBKYoz
OzzmqYZPAgMBAAECggEAFvXkdRa587KFZGRqhgZXydPElL94X6DLemFAzKi93abe
zeINsPmUPIi3C52iq7OlcYO5G6B4BZoVjSqjLh2UajxDPGzuHHo0eIhyQ/tl1Uia
m6CuhXp2uunNghFMd5C2+7IF5Ha/7lMrZbErvZmk6HxBaOhCVvaoOiJHHtiRV0aQ
3nYd2tqcAUyMaP05BMIvmfJMY1dpKxEYEe6ISpliUZ+m+GOAz8Wh3gJsmB4qQD5y
mXiBJnt/ZICm34vMNlh8Frk8Y4rNjzW/NuzOGUKc2uE+NM7ZMKjE9fJ+59cRr2mL
pT8AnfVoaJPn+5vGYO9QqN5b0PsJOJR62UAe1OAmIQKBgQD88z7VPCHbpGt3Qf1P
vcW4WSB+dZRvKwnrzQXWdduJHgwXgL6XM835i9OcDSIzH2/XKqLqbM7Q4tMdBoEV
W66B482SJYIO/bs8fTWKvmaclkssOZY3zE+ssCy6OHUUsTjjP/Aa1XYHgYfuXpo8
tQl8kGOarIbNpIEbCgryADt2UQKBgQDtP/HYPlNE+SLx3QgCaUM98nuesa4KJkX1
xTPgW/WWSGKmgYA/8yodWB54/uoaE96r90utgR7kD/EqIZNPIORKrwdF3Hoyp2cy
```

```
NEwbHhfycLdLWMnT3uHXhHe3bHi1oH2OJb4MwBbXXfzT1aqaJg/VkCGpQ/RwV7wJ
87+fmAXqnwKBgQCNTkmkj3IOpFxQg/HBkS8aG9gWLRMdwzpImwEO3Eom4DO7B/Xw
u8TuChnNjFy1mHM/DRdzxBMJCB2NkmucBdqHvz3AzelI/J+TXBMEDfd+tEsMSqNL
Snyb6NMjuJRXWeWAteAo7WSBLvBDzl91WGETUAoEhyrLSIHDHCofJNgS8QKBgFQ+
O3sEjn7UVEM7sAdjJzxM6PZtsxXphzgA4S3OmYLWiykCnO6YQqEO1Cs0oR3HZm77
rcSbL8PdoU3LhEo6hJpcUiHNaxqndK5QAaIzal9tBLjdezUfVGQ5pmTcQHSF23zn
VfejEblBp2/OO9JOp5q2xi0bcyWdEakbAzUBPOCTAoGAZ2PWduB8LWZ5FQw4JTov
rwXTulGRJRGhh4/PuAQ7UlvxzbSmXaetyiASy0z9Svqj1RGSdrlQG5+PtOpBqKZB
7DB2gwSC/bwMZ1Sr4G6F7VWLJyb7BIH2dXmQEGq6taGyCiOju7IogZ3+6TPikpRF
O2vjBR9QTLrMlj0H1y6Dz9c=
-----END PRIVATE KEY-----
```

注：这里只对公钥部分进行处理

3.2.2 运行结果

这里程序的输出结果的输出顺序与 `1.2 证书组成结构` 部分顺序相同：

```
版本号: 3
序列号: 1
签名算法: SHA1withRSA
颁发者: C=RU + CN=Test
有效起始日期: Mon Feb 01 00:00:00 CST 2016
有效终止日期: Fri Feb 01 00:00:00 CST 2019
主体名: C=RU + CN=Test
签名: [B@5c647e05
公钥:
48, -126, 1, 34, 48, 13, 6, 9, 42, -122, 72, -122, -9, 13, 1, 1, 1, 5, 0, 3, -126, 1, 15, 0, 48, -126, 1, 10, 2, -126, 1, 1, 0, -22, 108, 95, -2, -120, 30, 22, 110, -84, -
, 5, -44, 114, 31, 114, -104, 116, -13, 119, -93, -98, 3, 20, -24, 9, 33, -122, 14, 37, 46, 91, 26, 43, 9, -27, -77, -11, -79, -108, 41, -72, 107, 43, -1, 55, 88, -12
, 39, 40, -50, -32, -98, 126, -15, 74, -64, -49, 21, 3, 46, -51, 108, -71, 120, -88, -128, 0, -27, -107, -120, 40, -92, -77, 46, -71, 88, -40, -38, 3, 90, -17, -61,
3, 45, 5, 106, -123, -3, -33, -5, 50, 47, -114, -105, -4, 2, -53, 119, -63, 55, -108, -101, 99, -21, -107, -93, -73, 66, 117, 5, 43, -29, -47, 59, -1, 35, 22, 63, -6
, -13, 1, 11, -52, -9, -100, 66, 113, 86, 51, -114, 96, -84, 84, 27, -9, 98, -46, -40, 124, 120, 121, -31, 99, 103, 103, -101, 86, 49, 98, -93, 71, 104, 117, 71, 83,
8, 122, 13, 82, -42, 15, -59, -34, -22, -125, 32, 24, 26, -63, 41, -118, 51, 59, 60, -26, -87, -122, 79, 2, 3, 1, 0, 1,
```

- 版本号：3
- 序列号：1
- 签名算法：SHA1withRSA
- 颁发者：C = RU + CN = Test
- 有效起始日期：Mon Feb 01 00:00:00 CST 2016
- 有效终止日期：Fri Feb 01 00:00:00 CST 2019
- 主体名：C=RU + CN=Test
- 签名：[B@5c647e05
- 公钥：
  48,-126,1,34,48,13,6,9,42,-122,72,-122,-9,13,1,1,1,5,0,3,-126,1,15,0,48,-126,1,10,2,-126,1,1,0,-22,108,95,-2,-120,30,22,110,-84,-51,-77,-97,-107,120,-115,-4,59,-62,72,-86,-108,21,-107,41,83,5,-44,114,31,114,-104,116,-13,119,-93,-98,3,20,-24,9,33,-122,14,37,46,91,26,43,9,-27,-77,-11,-79,-108,41,-72,107,43,-1,55,88,-121,15,109,67,66,89,-67,127,-73,-106,110,107,-12,-55,5,54,8,62,39,40,-50,-32,-98,126,-15,74,-64,-49,21,3,46,-51,108,-71,120,-88,-128,0,-27,-107,-120,40,-92,-77,46,-71,88,-40,-38,3,90,-17,-61,-107,-5,67,-111,124,-22,-79,-28,95,72,54,-112,-61,-107,72,123,45,5,106,-123,-3,-33,-5,50,47,-114,-105,-4,2,-53,119,-63,55,-108,-101,99,-21,-107,-93,-73,66,117,5,43,-29,-47,59,-1,35,22,63,-69,-107,-19,40,-13,-46,58,36,62,-57,-109,15,-3,-71,65,-76,113,-13,1,11,-52,-9,-100,66,113,86,51,-114,96,-84,84,27,-9,98,-46,-40,124,120,121,-31,99,103,103,-101,86,49,98,-93,71,104,117,71,83,61,54,-99,-121,-109,-101,90,-60,-46,-31,79,-83,-65,103,-19,28,122,13,82,-42,15,-59,-34,-22,-125,32,24,26,-63,41,-118,51,59,60,-26,-87,-122,79,2,3,1,0,1,

4. 参考文献

- 维基百科-X509证书
- X509补充
- Java操作X509的方法
- CertificateFactory方法
- Java的异常处理

---

5. 代码

Github

6.个人博客

CSDN