# java生成数字证书(3)

原文链接：https://blog.csdn.net/happylee6688/article/details/42266465

## 序

前两篇把基本的概念和构成都大致的说了一下，今天这篇文章，主要是讲一下，如何使用 Java 代码生成 CA 证书，以及在生成证书的时候，需要设置的一些属性。

## 正文

废话不多说，直接上内容。

这里使用的是 Java 的 api ，以及第三方的一个组件 —— BC，（Bouncy Castle）。稍微介绍一下 BC，Bouncy Castle 是一种用于 Java 平台的开放源码的轻量级密码术包。它支持大量的密码术算法，并提供 JCE 1.2.1 的实现。而我们将要使用的就是非常常用的非对称算法 RSA 加密算法。

下面我们来看一下具体的代码。

## CAConfig （配置接口）

|  |
| --- |
| <span style="font-family:Comic Sans MS;font-size:12px;">package com.cacss.jsceu.context;    /\*\*  \* Created With IntelliJ IDEA.  \*  \* @author : lee  \* @group : sic-ca  \* @Date : 2014/12/30  \* @Comments : 配置接口  \* @Version : 1.0.0  \*/  public interface CAConfig {    /\*\*  \* C  \*/  String CA\_C = "CN";  /\*\*  \* ST  \*/  String CA\_ST = "BJ";  /\*\*  \* L  \*/  String CA\_L = "BJ";  /\*\*  \*/  String CA\_O = "SICCA";    /\*\*  \* CA\_ROOT\_ISSUER  \*/  String CA\_ROOT\_ISSUER="C=CN,ST=BJ,L=BJ,O=SICCA,OU=SC,CN=SICCA";  /\*\*  \* CA\_DEFAULT\_SUBJECT  \*/  String CA\_DEFAULT\_SUBJECT="C=CN,ST=BJ,L=BJ,O=SICCA,OU=SC,CN=";    String CA\_SHA="SHA256WithRSAEncryption";    }</span> |

## BaseCert （证书类）

|  |
| --- |
| <span style="font-family:Comic Sans MS;font-size:12px;">package com.cacss.jsceu.test;    import com.cacss.jsceu.context.CAConfig;  import com.cacss.jsceu.util.CertUtil;  import com.cacss.jsceu.util.DateUtil;  import org.bouncycastle.jce.provider.BouncyCastleProvider;  import org.bouncycastle.x509.X509V3CertificateGenerator;    import javax.security.auth.x500.X500Principal;  import java.security.\*;  import java.security.cert.X509Certificate;      /\*\*  \* Created With IntelliJ IDEA.  \*  \* @author : lee  \* @group : sic-ca  \* @Date : 2014/12/30  \* @Comments : 证书类  \* @Version : 1.0.0  \*/  @SuppressWarnings("all")  public class BaseCert {  /\*\*  \* BouncyCastleProvider  \*/  static {  Security.addProvider(new BouncyCastleProvider());  }  /\*\*  \*  \*/  protected static KeyPairGenerator kpg = null;    /\*\*  \*  \*/  public BaseCert() {  try {  // 采用 RSA 非对称算法加密  kpg = KeyPairGenerator.getInstance("RSA");  // 初始化为 1023 位  kpg.initialize(1024);  } catch (NoSuchAlgorithmException e) {  e.printStackTrace();  }    }    /\*\*  \* 生成 X509 证书  \* @param user  \* @return  \*/  public X509Certificate generateCert(String user) {  X509Certificate cert = null;  try {  KeyPair keyPair = this.kpg.generateKeyPair();  // 公钥  PublicKey pubKey = keyPair.getPublic();  // 私钥  PrivateKey priKey = keyPair.getPrivate();  X509V3CertificateGenerator certGen = new X509V3CertificateGenerator();  // 设置序列号  certGen.setSerialNumber(CertUtil.getNextSerialNumber());  // 设置颁发者  certGen.setIssuerDN(new X500Principal(CAConfig.CA\_ROOT\_ISSUER));  // 设置有效期  certGen.setNotBefore(DateUtil.getCurrDate());  certGen.setNotAfter(DateUtil.getNextYear());  // 设置使用者  certGen.setSubjectDN(new X500Principal(CAConfig.CA\_DEFAULT\_SUBJECT + user));  // 公钥  certGen.setPublicKey(pubKey);  // 签名算法  certGen.setSignatureAlgorithm(CAConfig.CA\_SHA);  cert = certGen.generateX509Certificate(priKey, "BC");  } catch (Exception e) {  System.out.println(e.getClass() + e.getMessage());  }  return cert;  }  }</span> |

## GenerateCa （测试类）

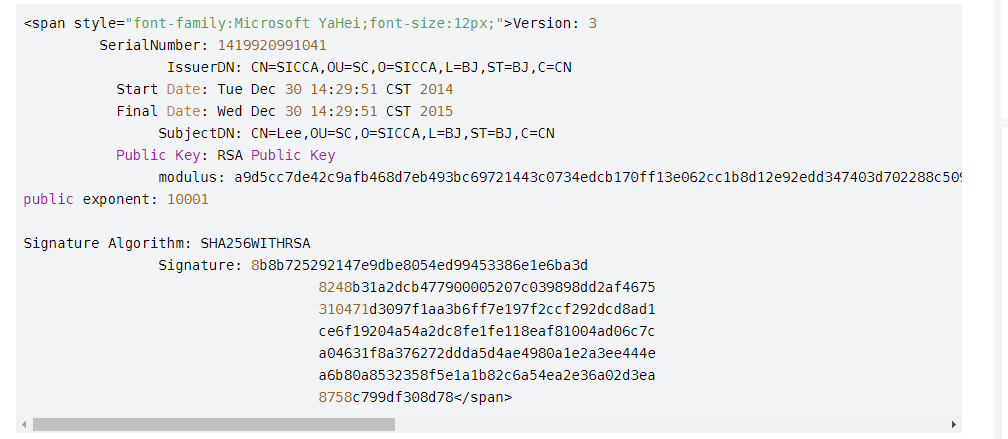
|  |
| --- |
| <span style="font-family:Comic Sans MS;font-size:12px;">package com.cacss.jsceu.test;    import java.io.FileNotFoundException;  import java.io.FileOutputStream;  import java.io.IOException;  import java.security.cert.CertificateEncodingException;  import java.security.cert.X509Certificate;    /\*\*  \* Created With IntelliJ IDEA.  \*  \* @author : lee  \* @group : sic-ca  \* @Date : 2014/12/30  \* @Comments : 测试证书类  \* @Version : 1.0.0  \*/  public class GenerateCa {  private static String certPath = "d:/lee.cer";  public static void main(String[] args) {  BaseCert baseCert = new BaseCert();  X509Certificate cert = baseCert.generateCert("Lee");  System.out.println(cert.toString());    // 导出为 cer 证书  try {  FileOutputStream fos = new FileOutputStream(certPath);  fos.write(cert.getEncoded());  fos.close();  } catch (FileNotFoundException e) {  e.printStackTrace();  } catch (CertificateEncodingException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }  }  }</span> |

## 效果图

下面是生成的证书，以及导出为 cer 格式的证书。

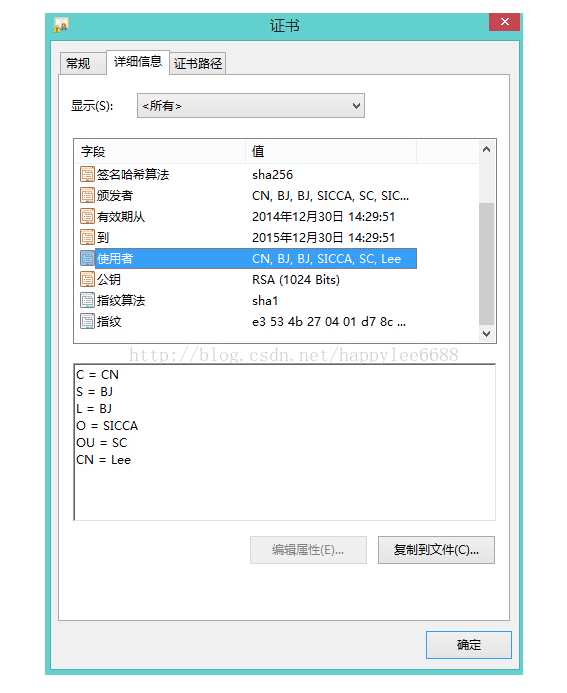
## 控制台打印

|  |
| --- |
| <span style="font-family:Microsoft YaHei;font-size:12px;">Version: 3  SerialNumber: 1419920991041  IssuerDN: CN=SICCA,OU=SC,O=SICCA,L=BJ,ST=BJ,C=CN  Start Date: Tue Dec 30 14:29:51 CST 2014  Final Date: Wed Dec 30 14:29:51 CST 2015  SubjectDN: CN=Lee,OU=SC,O=SICCA,L=BJ,ST=BJ,C=CN  Public Key: RSA Public Key  modulus: a9d5cc7de42c9afb468d7eb493bc69721443c0734edcb170ff13e062cc1b8d12e92edd347403d702288c5094ef2d0b2e811e0ee779a5e0a0cb7d5c75f30c5063eaa87aae7ba06bb3cf6ce6b0a5b0cd0cc2756255aff91fb09266b5dbbb6af491b5313947529d6a1fc30b9407ba1059bae909226c34e196b53c757a5826ffe147  public exponent: 10001  Signature Algorithm: SHA256WITHRSA  Signature: 8b8b725292147e9dbe8054ed99453386e1e6ba3d  8248b31a2dcb477900005207c039898dd2af4675  310471d3097f1aa3b6ff7e197f2ccf292dcd8ad1  ce6f19204a54a2dc8fe1fe118eaf81004ad06c7c  a04631f8a376272ddda5d4ae4980a1e2a3ee444e  a6b80a8532358f5e1a1b82c6a54ea2e36a02d3ea  8758c799df308d78</span> |



## cer 证书





## 结束语

我这里使用的是第三方的组件 BC 包进行加密的，采用的是 RSA 的加密算法，证书中的密钥长度为 1024 位，当然，你也可以设置为 2048 位，根据你自己的需要选择即可。不过，需要说明的一点是，在使用公（私）钥加密的时候，需要加密的字符串的长度是有要求的，以 1024 位的密钥长度来说，那么需要加密的字符串的长度不能超过 117 个字符，计算公式：1024 / 8 - 11 = 117。所以，在加密长字符串的时候，就需要采用分片加密的方法了，这一点需要注意，当然，后续讲加密的文章中，我也会说明这一点。