公告

昵称：FangJinuo
园龄：6年
粉丝：223
关注：12
+加关注

< 2019年4月 >

日	一	二	三	四	五	六
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

搜索

找找看

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

最新随笔

1. Java监控类库Metrics
2. Python：decorator [转]
3. Python 提案
4. JVM：GC
5. Python：import 与__import__()
6. Python：字符串格式化
7. Python： 遍历
8. Windows系统资源监控
9. IO Redirect 与 Pipe
10. Windows cmd

随笔分类(342)

Android(3)
Ant(3)
AOP(5)
Apache Commons(3)
APM(1)
BPMN
ByteCode(6)
C、C++(3)
Cache(7)
CAS(2)
docker(1)
Dojo(1)
ElasticSearch
FreeMarker(1)
Git(3)
GlassFish(2)
Gradle(1)
Hadoop(3)
HBase(2)
Hibernate(3)
HTML、HTML5、CSS(9)
J2EE(15)
J2SE(49)
Java XML(2)
JavaScript(18)
jQuery(6)
JSR(3)
JTA(1)
Kafka(7)
Linux(16)
Logger(7)
Maven(2)
MQ、JMS(2)
MyBatis(3)
Netty、Mina、Grizzly(8)
Python(14)
Redis
Scala(1)
Security(8)
Selenium、WebDriver(2)
Spark(1)
Spring(28)
SQL(14)
Struts2(1)
TCP/IP(1)
Test(4)
Thrift、Protobuf(1)
Tomcat(18)
WebLogic(2)
WebService(3)
Windows(3)
YAML(1)
ZooKeeper(5)
分布式(4)
工具(12)
设计模式(7)
事务(4)
项目管理(1)
杂文(9)

随笔档案(258)

2019年2月 (1)
2018年11月 (3)
2018年9月 (3)
2018年8月 (3)
2018年7月 (1)
2018年6月 (4)
2018年5月 (4)
2018年4月 (5)
2018年3月 (10)
2018年2月 (1)
2018年1月 (6)

Java Security：Java加密框架(JCA)简要说明

加密服务总是关联到一个特定的算法或类型,它既提供了密码操作(如Digital Signature或MessageDigest),生成或供应所需的加密材料(Key或Parameters)加密操作,也会以一个安全的方式生成数据对象(KeyStore或Certificate),封装(压缩)密钥(可以用于加密操作)。

Java Security API中，一个engine class就是定义了一种加密服务，不同的engine class提供不同的服务。下面就来看看有哪些engine class：

1）MessageDigest：对消息进行hash算法生成消息摘要（digest）。

2）Signature：对数据进行签名、验证数字签名。

3）KeyPairGenerator：根据指定的算法生成配对的公钥、私钥。

4）KeyFactory：根据Key说明（KeySpec）生成公钥或者私钥。

5）CertificateFactory：创建公钥证书和证书吊销列表（CRLs）。

6）KeyStore：keystore是一个keys的数据库。Keystore中的私钥会有一个相关联的证书链，证书用于鉴定对应的公钥。一个keystore也包含其它的信任的实体。

7）AlgorithmParameters：管理算法参数。KeyPairGenerator就是使用算法参数，进行算法相关的运算，生成KeyPair的。生成Signature时也会用到。

8）AlgorithmParametersGenerator：用于生成AlgorithmParameters。

9）SecureRandom：用于生成随机数或者伪随机数。

10）CertPathBuilder：用于构建证书链。

11）CertPathValidator：用于校验证书链。

12）CertStore：存储、获取证书链、CRLs到（从）CertStore中。

从上面这些engine class中，可以看出JCA（Java加密框架）中主要就是提供了4种服务：Digest、Key、Cert、Signature、Alogorithm。

1） 对消息内容使用某种hash算法就可以生成Digest。

2） 利用KeyFactory、KeyPairGenerator就可以生成公钥、私钥。

3） 证书中心使用公钥就可生成Cert。

4） 可以使用私钥和Digest就可以消息进行签名Signature。

5） 不论是Digest、Key、Cert、Signature，都要使用到算法Algorithm。

JCA Core API

1）engine class的提供商Provider

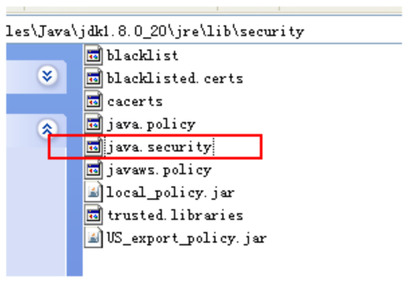
从JCA的设计上来说，这些engine的实现都离不开Provider。

这个类继承了Properties，提供了JCA中的engine class。每个engine class都有getInstance()方法，它们都是从provider中获取相关实例的。所以说Provider是JCA engine class的提供商。

2）管理Provider的工具：Security

其实就是一个存放Provider的集合。如果你自定义了一个Provider，可以使用Java Security属性文件配置provider，也可以直接使用Security采用编程的方式来添加Provider。然后就可以使用自定义的engine class了。

Java Security 属性文件在Java Security Policy中已有提过。在安装目录下：



下面是一个自定义的Provider：

```
/**
 * @author fs1194361820@163.com
 */
public class XYZProvider extends Provider{
    public XYZProvider(){
        super("XYZ", 1.0, "XYZ Security Provider v1.0");
        put("MessageDigest.XYZ", XYZMessageDigest.class.getName());
    }
}
```

已经默认配置了下列Provider：

```
security.provider.1=sun.security.provider.Sun
security.provider.2=sun.security.rsa.SunRsaSign
security.provider.3=sun.security.ec.SunEC
security.provider.4=com.sun.net.ssl.internal.ssl.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
security.provider.7=com.sun.security.sasl.Provider
security.provider.8=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.9=sun.security.smartcardio.SunPCSC
security.provider.10=sun.security.mscapi.SunMSCAPI
```

配置为：security.provider.11=com.fjn.security.XYZProvider 即可。

编码方式就更加简单了：Security.*addProvider*(**new** XYZProvider());

3）消息摘要服务：MessageDigest

消息摘要服务其实就是使用hash算法将一段消息（可以是字符串、文件内容、html等）进行计算生成的一个byte[]。

常用加密算法MD5、SHA、SHA-1其实都是hash算法。

下面就给一个简单的MD5算法工具：

```
View Code
```

Md5算法我并没有去实现，因为在JDK中已经内置了md5算法。上面的代码就是使用消息摘要服务，并使用md5算法，生成相应的摘要。

下面来一个自定义的MessageDigest：

```
View Code
```

2017年12月 (8)
2017年11月 (2)
2017年10月 (1)
2017年9月 (3)
2017年8月 (8)
2017年7月 (3)
2017年6月 (3)
2017年5月 (2)
2017年4月 (3)
2017年3月 (6)
2017年2月 (2)
2017年1月 (5)
2016年12月 (13)
2016年11月 (4)
2016年10月 (4)
2016年8月 (2)
2016年7月 (4)
2016年6月 (5)
2016年5月 (7)
2016年4月 (9)
2016年3月 (2)
2016年1月 (5)
2015年12月 (10)
2015年11月 (1)
2015年10月 (3)
2015年9月 (5)
2015年8月 (4)
2015年7月 (7)
2015年6月 (8)
2015年5月 (6)
2015年4月 (3)
2015年3月 (3)
2015年2月 (3)
2015年1月 (2)
2014年12月 (3)
2014年11月 (1)
2014年10月 (3)
2014年9月 (8)
2014年8月 (22)
2014年7月 (14)
2014年6月 (1)
2014年4月 (1)
2013年9月 (1)
2013年8月 (7)

文章档案(1)
2014年8月 (1)

积分与排名
积分 - 157939
排名 - 2478

最新评论
1. Re:FindBugs 入门——帮你减少代码中的bug数
你好，请问findbugs能实现自动化吗？要检测的项目数量比较大
--baopanpan
2. Re:Spring源码阅读：使用标准AOP的API模拟Spring AOP + AspectJ的设计与实现
你好,楼主你写的文章很好,我收益匪浅,非常感谢,但你这篇中的AbstractMethodInterceptor 中invoke(MethodInvocation invocation)似乎在这里没有用.....
--脑子是个好东西
3. Re:Kafka: LogManager
楼主转发一下
--你知道所有的未来
4. Re:EJB：快速入门
@XiaoLeng没关系，问题解决了就好...
--FangJinuo
5. Re:EJB：快速入门
@FangJinuo解决了 不好意思了~...
--XiaoLeng
6. Re:EJB：快速入门
@XiaoLeng现在问题解决了吗...
--FangJinuo
7. Re:EJB：快速入门
@
不好意思回复错了。。。
--XiaoLeng
8. Re:EJB：快速入门
我照着您的教程一步一步建的项目，然后一直报错。。。最后发现您创建项目时候写的是EBJTest，但调用项目时候写的是EJBTest。。。建议您修改一下。。避免有像我这样的小白遇到问题查了好久都查不出来。.....
--XiaoLeng
9. Re:Python：Class
你好！我是图书编辑，有兴趣写一本python相关的书么？13647276727这是我微信号，有兴趣可以加下我。
--图书策划编辑
10. Re:Spring源码阅读：使用标准AOP的API模拟Spring AOP + AspectJ的设计与实现
@从精于一开始谬赞了，哈哈...
--FangJinuo

阅读排行榜
1. ZooKeeper：第三方客户端ZKClient(19336)
2. Java Security：Java加密框架(JCA)简要说明(11033)
3. 使用 Eclipse 玩转 C、C++(10777)
4. FindBugs 入门——帮你减少代码中的bug数(9670)
5. Spring源码阅读：IOC容器的设计与实现（二）——ApplicationContext(8490)
6. Mina 快速入门(6902)

这个自定义的MessageDigest中备注的内容，其实就是MessageDigest的执行过程，所有的MessageDigest都要遵从这个过程的。

测试用例：

View Code

在这个用例中，writeMessage()将一段字符串保存后并将生成的digest也保存。

readMessage()将消息读取后，使用MessageDigest.isEqual()方法进行比较，这样可以知道文件是否被人改动过。

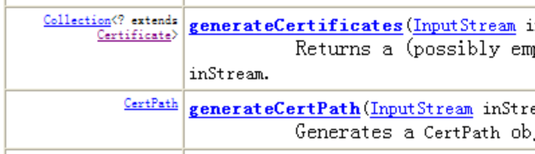
而实际上利用私钥更新签名信息时，就是使用MessageDigest#update()方法的。

4）Key 相关的服务

Key包括公钥（PublicKey）、私钥（PrivateKey）两种。

4.1 KeyPairGenerator

这个服务用于生成PublicKey和PrivateKey。



获取实例后，只需要根据上面4种initialize方法进行初始化后，就可以生成KeyPair了。

View Code

第一次调用generateKeyPair()都会生成不同的KeyPair。KeyPairGenerator 每次生成的都是一个KeyPair。

4.2 KeyFactory

KeyFactory用于在Key与KeySpec之间转换，即可以根据key获取到KeySpec，也可以根据KeySpec获取Key。

View Code

5）Cert相关的服务

从上一篇的例子中知道，用户使用的Public Key有可能被不法分子偷偷地窜改，这样用户就得不到应有的服务，也会受到不法分子的危害。如何保证public key不被窜改或者替换呢？认证服务就出现了。

5.1 CertificateFactory

用于生成Certificate或者CRL的。

Certificate	**generateCertificate**(InputStream inStream) Generates a certificate object and initializes it
Collection<? extends Certificate>	**generateCertificates**(InputStream inStream) Returns a (possibly empty) collection view of the inStream.
CertPath	**generateCertPath**(InputStream inStream) Generates a CertPath object and initializes it wit
CertPath	**generateCertPath**(InputStream inStream, String encoding) Generates a CertPath object and initializes it wit
CertPath	**generateCertPath**(List<? extends Certificate> certificates) Generates a CertPath object and initializes it wit
CRL	**generateCRL**(InputStream inStream) Generates a certificate revocation list (CRL) ob input stream inStream.
Collection<? extends CRL>	**generateCRLs**(InputStream inStream) Returns a (possibly empty) collection view of the
Iterator<String>	**getCertPathEncodings**()

```
FileInputStream fis = new FileInputStream(filename);
 BufferedInputStream bis = new BufferedInputStream(fis);

 CertificateFactory cf = CertificateFactory.getInstance("X.509");

 while (bis.available() > 0) {
    Certificate cert = cf.generateCertificate(bis);
    System.out.println(cert.toString());
 }
```

什么是CRL ?

一个证书颁发机构需要证书吊销其颁发的证书——也许是虚假的,或者证书的用户已经使用证书从事非法行为。在这样的情况下,证书的有效期不足保护;证书必须立即失效。

下面的这个例子就是在验证完证书的有效性后，判断这个证书是否是一个吊销的证书。

View Code

5.2 CertPathBuilder构建证书链CertPath

CertPath就是之前说的证书链。其实就是一个Certificate的有序列表。在列表的最后的一个Cert是一个自签名的Cert。

5.3 CertPathValidator验证Cert链

CertPathValidator用于校验Cert。

6）KeyStore

一个KeyStore是一个key、cert的库，里面存储了PrivateKey, Aliases, Certs.

KeyStore将会有专门的说明。

7）Signature签名

用私钥签名，用公钥验证；

7. Tomcat：利用Apache配置反向代理、负载均衡(6859)
8. Java Se：Java Security(6328)
9. MyBatis：打印SQL 日志(6168)
10. JQuery插件：遮罩+数据加载中。。。（特点：遮你想遮，罩你想罩)(5791)

评论排行榜

1. Mina 快速入门(10)
2. 模拟JavaEE的Filter(10)
3. JavaSE：你真的了解继承、重写、可见性吗？(9)
4. Ajax跨域访问(9)
5. EJB：快速入门(8)
6. C++： 主要知识点(7)
7. Git 入门(5)
8. Android APK 反编译(5)
9. Spring源码阅读：使用标准AOP的API模拟Spring AOP + AspectJ的设计与实现(5)
10. Java Annotation 学习(5)

推荐排行榜

1. JQuery插件：遮罩+数据加载中。。。（特点：遮你想遮，罩你想罩)(8)
2. Kafka：架构简介(8)
3. Spring源码阅读系列总结(8)
4. JavaSE：你真的了解继承、重写、可见性吗？(7)
5. 深入学习Java8 Lambda （default method, lambda, function reference, java.util.function 包）(6)
6. 把你的Project发布到GitHub上(5)
7. Tomcat源码解读：我们发起的HTTP请求如何到达Servlet的(5)
8. Tomcat源码解读：Tomcat启动过程都干了啥(4)
9. Git 入门(4)
10. SLF4j：Log facade abstract(3)

View Code

到此，JCA部分的engine class已经大体上有个了解了。接下来就是要学习如何应用它们了。

分类：J2SE,Security

标签：Java Security, Security, PublicKey, PrivateKey, Certificate, MessageDigest, KeyStore, KeyFactory, Signature, JCA

« 上一篇：Java Security：公钥私钥、数字签名、消息摘要是什么
» 下一篇：Java Security：keytool工具使用说明

posted @ 2015-01-30 15:40 FangJinuo 阅读(11033) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

相关博文：
· Java jvisualvm简要说明
· java jvisualvm简要说明
· Java jvisualvm简要说明
· UIKit框架各类简要说明
· Java安全体系（JCA）分析

最新新闻：
· 电动自行车充电服务运营商「小绿人科技」完成亿元B轮融资，即将启动本地化服务
· 一周三休？ 日本微软对员工作息制度进行探索
· 中科大潘建伟团队刷新量子多体纠缠态纪录
· AI 的主打歌：程序员打得作曲家神不守舍
· 最前线丨滴滴首次对外公布网约车平均抽成19%，新业务导致亏损压力变大
» 更多新闻...