

## Pràctica 4: *Certificats Digitals.*

### Objectiu

L'objectiu d'aquesta pràctica és tractar amb certificats digitals. Les eines que proporciona el llenguatge java per a l'ús de certificats són bastant limitades. Per això utilitzarem una llibreria externa [2]. Una vegada vistos els certificats, veurem com crear repositoris per emmagatzemar claus i certificats de forma homogènia i segura.

L'aplicació resultant la utilitzarem en la última de les pràctiques en què treballarem amb protocols criptogràfics.

### Certificats Digitals i IAIK

Com ja hem introduït anteriorment el llenguatge Java no ofereix eines que permetin tractar i crear de forma completa els certificats digitals. Per això, en aquesta pràctica utilitzarem una llibreria externa que permet crear certificats digitals, que després podran utilitzar-se per interactuar amb altres entitats criptogràfiques del llenguatge.

Tractarem els certificats com entitats que construïm a partir de la clau pública d'una persona. A més, afegirem altres dades com la data de caducitat, el nom del propietari o bé la signatura digital de l'emissor del certificat.

El tipus de certificats amb els que treballarem són els que segueixen l'estàndard del IEEE X.509. La llibreria IAIK proporciona la classe *iaik.x509.X509Certificate* que permet crear certificats i tractar certificats ja creats. Algunes de les funcions més importants són les següents:

- *setSubjectDN(Principal name)* Afegeix el nom del propietari del certificat.
- *setIssuerDN(Principal name)* Afegeix el nom del signant del certificat al certificat.
- *setPublicKey(PublicKey pk)* Afegir la clau pública del propietari dins el certificat.
- *setValidNotAfter(Date d)* Posar data de validesa màxima a aquest certificat.
- *setValidNotBefore(Date d)* Posar data de validesa mínima a aquest certificat.

- *setSerialNumber(BigInteger s)* Posar número de sèrie a aquest certificat.
- *sign(AlgorithmID alg, PrivateKey sk)* Signar aquest certificat utilitzant la clau privada corresponent.
- *verify(PublicKey p)* Verificar la signatura d'aquest certificat utilitzant la clau pública corresponent.
- *writeTo(OutputStream os)* Escriure aquest certificat en un stream, per exemple, un stream d'arxiu.

## Emmagatzemat de claus i Certificats

En les pràctiques anteriors ja hem vist algunes de les possibilitats que ens ofereix el llenguatge java alhora de emmagatzemar claus a disc. Tot i així, els sistemes vistos són més aviat rudimentaris i no proporcionen una manera estàndard i homogènia de emmagatzemar les nostres claus a disc de forma segura.

Una alternativa als sistemes vistos fins ara és utilitzar *Key Stores*. Aquestes entitats són repositoris que permeten emmagatzemar, en un sol arxiu, múltiples claus i/o certificats de forma segura.

Els *Key Stores* es manipulen a través de la classe *KeyStore*, un motor criptogràfic com els que hem vist en les pràctiques anteriors. Alguns dels mètodes més destacables d'aquesta classe són:

- *getInstance(String Algorithm)* Crea un objecte *KeyStore*. Els tipus especificats al camp algorisme poden ser "JKS", "JCEKS", o bé "PKCS12".
- *store(OutputStream os, char[] password)* Emmagatzema la *Key Store* en l'*OutputStream* especificat.
- *load(InputStream is, char[] password)* Carrega el contingut d'una *Key Store* que es llegeix a través de l'*stream* especificat.
- *setCertificateEntry(String alias, Certificate cert)* Afegeix un certificat al nostre *KeyStore*. Aquest certificat serà referenciat a través de l'*alias* que es passa per paràmetres.
- *setKeyEntry(String alias, Key key, char[] password, Certificate[] chain)* Afegeix una parella clau privada-certificat al *Key Store*. La clau privada és emmagatzemada xifrada utilitzant el password especificat. L'entrada corresponent també és referenciada a través d'un alias.

## Enunciat

En aquesta pràctica haureu de omplir les funcions buides que es troben dins l'arxiu *CertMaker.java*. Aquesta classe, controlarà un objecte *KeyStore* en el qual podrem emmagatzemar els certificats i claus que generem utilitzant la nostra aplicació. Les funcions que heu de omplir són les següents:

- *CertMaker(String file, String password)* El constructor de la classe rebrà el nom d'un arxiu en el que es troba emmagatzemat un Key Store. En cas de que estiguem creant un KeyStore de zero, el valor de file serà *null*. Al final de la construcció de l'objecte, s'haurà creat un objecte *KeyStore* amb l'arxiu introduït.
- *void save(String filename)* Emmagatzema els canvis fets sobre el Key Store en l'arxiu especificat.
- *X509Certificate makeSelfSigned(String alias, String password)* Crea un certificat autosignat i l'emmagatzema en el KeyStore juntament amb la clau creada per signar-lo.
- *X509Certificate makeSigned(String alias, String password, String aliassigner, String passwordsigner)* Crea un certificat signat amb alguna de les claus emmagatzemades dins el Key Store.
- *void importTrusted(String alias, String file)* Importar un certificat d'un arxiu i introduir-lo dins el Key Store amb un alias determinat.

## Referències

- [1] Java API. <http://java.sun.com/j2se/1.5.0/docs/api/>
- [2] IAIK API. [http://jce.iaik.tugraz.at/products/01\\_jce/documentation/javadoc/index.html](http://jce.iaik.tugraz.at/products/01_jce/documentation/javadoc/index.html)
- [3] Java Cryptography Architecture. <http://java.sun.com/j2se/1.5.0/docs/guide/security/CryptoSpec.html>.
- [4] Programació en java. <http://java.programacion.net>