Guiões das Aulas Práticas

Segurança Informática

Licenciatura em Tecnologias da Informação

Hélder Gomes

Escola Superior de Tecnologia e Gestão de Águeda
Universidade de Aveiro

2018–2019

Conteúdo

2

Assinaturas digitais com o Cartão de Cidadão em Java

Resumo:

- Assinaturas digitais.
- Dispositivos PKCS#11
- Assinaturas digitais com o Cartão de Cidadão.

2.1 Introdução

Para a realização deste trabalho é necessário ter um JDK (Java Development Kit) com versão superior à 7 instalado no seu computador. Caso não o tenha instalado, pode obtê-lo na página oficial de distribuição da versão SE (*Standard Edition*) do Java[1].

Poderá também ter instalado um IDE da sua preferência, como o NetBeans[2], por exemplo.

Também poderá ser útil visualizar o conteúdo de ficheiros em binário. Para esse efeito, se estiver em ambiente Linux, pode instalar a aplicação `okteta` (disponível no repositório de aplicações do Ubuntu). Caso esteja em ambiente Windows, existem vários disponíveis online, como o WinHex, por exemplo.

2.2 Software do Cartão de Cidadão

Para a realização deste trabalho é ainda necessário instalar o software do Cartão de Cidadão. Para isso é necessário primeiro descarregá-lo, o que pode fazer do respetivo sítio (`www.cartaodecidadao.pt`), tendo o cuidado de escolher uma versão adequada ao sistema operativo onde o vai instalar.

Faça duplo click sobre o ficheiro descarregado e siga as instruções para a instalação.

2.3 Norma PKCS#11

A norma PKCS#11 (Cryptographic Token Interface Standard) foi produzida pela RSA Security e define interfaces programáticas para dispositivos criptográficos, como *smartcards*, de que é exemplo o Cartão de Cidadão. O Java inclui um fornecedor (*provider*) de serviços de segurança que concretiza o modelo interação definido pelo PKCS#11, o Sun PKCS#11 (sun.security.pkcs11.SunPKCS11), que, em contraste com a maioria dos outros *providers*, não implementa ele próprio os algoritmos criptográficos. Ele actua como uma ponte entre a API criptográfica do Java (JCA) e e a API criptográfica PKCS#11 nativa, traduzindo chamadas e convenções entre os dois. Isto permite que aplicações Java, que usem a API do JCA, possam,

[1] `http://www.oracle.com/technetwork/java/javase/downloads/index.html`
[2] `http://www.netbeans.org`

sem modificações, tirar partido dos algoritmos criptográficos fornecidos pelos vários tokens que implementam a norma PKCS#11.
Nota: Uma forma alternativa de aceder às funcionalidades PKCS#11 é utilizando o IAIK PKCS#11 Wrapper. No entanto, este não é um provider de PKCS#11 de acordo com a arquitetura de criptografia do Java. O iaik PKCS#11 Wrapper pode ser obtido, após registo, da sua página na Web[3] ou, alternativamente, na área de software da página desta Unidade Curricular.

2.3.1 SUNPKCS11 *provider* e o Cartão de Cidadão

Para poder utilizar o Cartão de Cidadão, primeiro é necessário instalá-lo usando a JCA. Para isso é preciso criar um ficheiro de configuração onde se dá um nome ao *provider* e o caminho completo para a biblioteca do Cartão de Cidadão com a implementação da norma PKCS#11.

Consequentemente, deve começar por gerar um ficheiro de configuração com um nome e localização à sua escolha, e com o seguinte conteúdo (para sistemas Linux):

```
name=CartaoCidadao
library=/usr/local/lib/libpteidpkcs11.so
```

Se realizar este trabalho em ambientes Windows o caminho e o nome desta biblioteca são diferentes:

```
name=CartaoCidadao
library=C:\Windows\system32\pteidpkcs11.dll
```

O *provider* do Cartão de Cidadão pode ser instalado de forma estática ou dinâmica. Para o instalar de forma estática, deve adicioná-lo no ficheiro de configuração da segurança do Java (`$JAVA_HOME/lib/security/java.security`, em que `JAVA_HOME` é a pasta `jre` dentro da pasta do JDK que está a usar).Isso é feito adicionando, imediatamente a seguir ao último *provider* listado em `java.security`, uma linha semelhante ao seguinte fragmento, que instala o SunPKCS#11 com o ficheiro de configuração /home/user/pkcs11cc.cfg (adapte o número do provider para ser o imediatamente a seguir ao número do último provider listado no seu ficheiro e adapte o nome e localização do ficheiro de configuração à sua situação específica):

```
# configuration for security providers 1-6 omitted
security.provider.7=sun.security.pkcs11.SunPKCS11 /home/user/pkcs11cc.cfg
```

[3] `http://jce.iaik.tugraz.at/sic/Products/Core-Crypto-Toolkits/PKCS_11_Wrapper`

NOTA: Caso esteja em abiente Windows, no ficheiro java.security, deve usar a barra (/) na indicação do caminho para o ficheiro de configuração (e.g., C:/Windows/).

2.4 Obter o *security provider* do Cartão de Cidadão

Crie um programa que liste todos os *security providers* suportados no seu sistema. Para o efeito inclua as seguintes linhas de código no seu projecto:

```
Provider[] provs = Security.getProviders();
for(int i = 0; i < provs.length; i++){
    System.out.println( i + " - Nome do provider: " + provs[i].getName() );
}
```

Compile e execute o seu programa e veja a lista de *security providers* obtida. Consegue identificar qual o *provider* do Cartão de Cidadão?

2.5 Conteúdo do Cartão de Cidadão

Para se poder realizar operações criptográficas utilizando o Cartão de Cidadão é necessário criar um `KeyStore` e iniciá-lo, o que pode se fazer utilizando o seguinte código (`prov` é uma variável com o *security provider* do Cartão de Cidadão):

```
KeyStore ks = KeyStore.getInstance( "PKCS11", prov );
ks.load( null, null );
```

Utilizando o `KeyStore` pode listar o conteúdo do Cartão de Cidadão (em termos de objetos PKCS#11). Para isso adicione o seguinte código:

```
Enumeration<String> als = ks.aliases();
while (als.hasMoreElements()){
    System.out.println( als.nextElement() );
}
```

A informação que obtém é a identificação de dois certificados de chave pública incluídos no Cartão de Cidadão. Através destes identificadores podemos identificar qual o certificado ou chave privada que pretendemos utilizar numa operação criptográfica.

2.6 Assinatura digital

Desenvolva um programa que seja capaz de criar uma assinatura digital de um documento, utilizando o Cartão de Cidadão, e guardá-la num ficheiro próprio, bem como guardar num outro ficheiro o certificado de chave pública correspondente à chave privada com que assinou o documento.

Sugestão: considere a utilização de objetos das classes **java.security.KeyStore** e **java.security.Signature**, e das interfaces **java.security.PrivateKey** e **java.security.Certificate**. Use o algoritmo `SHA256withRSA` para realizar a assinatura.

2.7 Validação da Assinatura

Desenvolva um programa que seja capaz de validar uma assinatura digital de um documento. O programa deve ter como entradas o ficheiro com a assinatura digital, o ficheiro com o certificado de chave pública do assinante e o ficheiro original (com o documento que foi assinado). Deve também mostrar no ecrã o nome da entidade que assinou o documento.

Sugestão: considere a utilização da interface **java.security.Certificate** e de objetos das classes **java.security.Signature**, **java.security.cert.X509Certificate** e **java.security.cert.CertificateFactory**.

2.8 Bibliografia

`http://docs.oracle.com/javase/tutorial/security/index.html`
`http://docs.oracle.com/javase/8/docs/technotes/guides/security/index.html` `http://docs.oracle.com/javase/tutorial/security/apisign/index.html` `https://mywiki.ncsa.illinois.edu/wiki/Java_PKCS11`