Pràctica T5S2P2. Generar i verificar la signatura digital

A la pràctica vorem com podem autenticar l'emissor i comprovar que el missatge no ha sigut alterat mitjançant la signatura digital.

1. Funcionament de la signatura digital

El funcionament de la signatura electrònica està basat en una clau pública i una clau privada.

L'emissor té un parell de claus: una s'utilitza per a xifrar i l'altra per a desxifrar.

L'emissor manté en secret la clau privada i posa a disposició del públic la clau pública.

L'emissor obté un resum del missatge a signar amb una funció hash (resum). El resum és una operació que es realitza sobre un conjunt de dades, de forma que el resultat obtés és un altre conjunt de dades de mida fixa, independentment de la mida original, i que té la propietat d'estar associat unívocament al missatge inicial, és a dir, és impossible tobar dos missatges distints que generen el mateix resultat a l'aplicar la funció hash.

L'emissor xifra el resum del missatge amb la clau privada. Ésta és la signatura electrònica que s'afegeix al missatge original.

El receptor, al rebre el missatge, obté de nou el seu resum mitjançant la funció hash. A més a més, desxifra la signatura utilitzant la clau pública de l'emissor obtenint el resum que l'emissor va calcular. Si ambdós coincideixen, la signatura és vàlida per la qual cosa compleix els criteris d'autenticitat i integritat, a més a més del de no repudi, ja que l'emissor no pot negar haver enviat el missatge que porta la seua signatura.

2. Desenvolupament de la pràctica

Per a crear una signatura digital necessitem un parell de claus (privada i pública) que ens serviran per a verificar l'autenticitat de la signatura.

El parell de claus es pot proporcionar a través de fitxers o poden ser generats pel propi programa. La classe KeyPairGenerator ens permet generar el parell de claus.

Primera fase: Creació de claus pública i privada

Utilitzarem el paquet java.security

```
import java.security.*;
```

Primerament, generem un objecte de tipus generador de claus (en este cas claus DSA)

```
KeyPairGenerator KeyGen = KeyPairGenerator.getInstance("DSA");
```

El següent pas és inicialitzar el generador de claus, utilitzem el mètode initialize() al que li passarem la mida de la clau (valor entre 512 i 1024 i que deu ser múltiple de 64) i un número aleatori.
Per generar el número aleatori utilitzem la classe SecureRandom que genera nombres pseudoaleatoris.

```
// INICIALITZE EL GENERADOR DE CLAUS
SecureRandom numero = SecureRandom.getInstance("SHA1PRNG");
KeyGen.initialize(1024, numero);
```

L'ultim pas és generar el parell de claus i emmagatzemar-les en objectes PrivateKey (clave privada) i PublicKey (clau pública):

```
// CREE EL PARELL DE CLAUS
KeyPair parell = GeneradorClaus.generateKeyPair();

// GUARDE LES CLAUS PÚBLICA I PRIVADA
PrivateKey clauPrivada = parell.getPrivate();
PublicKey clauPublica = parell.getPublic();
```

Segona fase: Signar les dades

Quan ja tenim creades les claus, es poden signar les dades. Per a fer això, utilitzarem una instància de la classe Signature. Un objecte d'esta classe es pot utilitzar per generar i verificar signatures.

Existeixen tres fases en l'ús de l'objecte Signature, ja siga per a signar o verificar les dades:

1. Inicialització
    a. Amb clau pública initVerify()
    b. Amb clau privada initSign()
2. Actualització update()
3. Signatura sign() ó Verificació verify ()

En el nostre exemple, creem un objecte Signature per crear o verificar signatures amb l'algorisme DSA (que es amb el que creem les claus).

```
Signature dsa = Signature.getInstance("SHA1withDSA");
```

Després de crear l'objecte, seguim les fases d'ús. Inicialitzem:

```
dsa.initSign(clauPrivada);
```

Actualitzem amb les dades que deuen signar-se:

```
// MISSATGE A SIGNAR
String missatge = "Mensaje a firmar";
dsa.update(missatge.getBytes());
```

Signem i obtenim dades signades en array de bytes:

```
// MISSATGE SIGNAT
byte[] signatura = dsa.sign();
```

Tercera fase: Verificar dades

En la tercera fase, el receptor deurà rebre el missatge i verificar la signatura digital utilitzant un objecte Signature. Utilitzarem l'algorisme DSA tal com vam fer al signar.

```
// RECEPTOR VERIFICA AMB CLAU PÚBLICA.....................
// Signature verificadsa = Signature.getInstance("MD5withRSA");
Signature verificadsa = Signature.getInstance("SHA1withDSA");
verificadsa.initVerify(clauPublica);
```

Finalment, utilitzarem el mètode update de la classe Signature amb les dades del missatge i el mètode verify per a comparar les dues signatures digitals. Si torna true, és l'original.

Entregable Pràctica U5P2

Donat el codi que es facilita, modificar-lo adequadament per a que realitze la generació, verificació de la signatura digital i mostre un missatge per pantalla indicant si s'ha verificat o no correctament.

Exemple d'execució esperada quan siga correcta la verificació:

```
<terminated> U5P2VerificaSignaturaDigital_S
Signatura verificada correctament
```