salesians
SARRIÀ

Seguretat i criptografia

Criptografia pràctica

Josep Gutiérrez

Departament d'informàtica
Salesians de Sarrià

Tècniques criptogràfiques

- Encriptació amb clau simètrica
- Encriptació amb clau pública
- Signatura digital de XML
- Verificació de signatures digitals en XML

Escenari Criptogràfic

- Crear un esquema criptogràfic no és una tasca trivial, i es sol fer combinant diferents tècniques per tal d'aprofitar la seguretat que aporta la criptografia de clau pública i la rapidesa de la criptografia simètrica.
- Un escenari habitual presenta els següents passos:
  - Cada part genera un parell de claus pública/privada utilitzant RSA.
  - Les parts s'intercanvien les seves claus públiques de manera segura
  - Cada part genera una clau secreta per al xifrat AES, i xifra la clau creada recentment utilitzant la clau pública RSA de l'altre.
  - Cada part utilitza la seva clau privada per desencriptar la clau AES encriptada anteriorment amb la clau pública RSA.
  - A partir d'aquí poden enviar i rebre dades de forma segura utilitzant la clau AES i criptografia simètrica.

Criptografia simètrica

- Un sistema de criptografia simètrica o de clau simètrica utilitza la mateixa clau per encriptar i desencriptar.
- Per a realitzar encriptació amb clau simètrica el sistema criptogràfic més utilitzat és **AES**.
- L'Advanced Encryption Standard (**AES**), també conegut com a **Rijndael** és un esquema de xifrat per blocs adoptat com un estàndard de xifrat pel govern dels Estats Units.
- AES utilitza una **Key** per a xifrar el missatge i un **IV** (initialization vector)
- La **key** és l'element principal que permet xifrar el missatge i ha de restar secreta en tot moment ja que la seva pèrdua podria comprometre la seguretat de tot el sistema
- El **IV** o vector d'inicialització no és necessari que sigui secret.

IV o vector d'inicialització

- Un **IV** o vector d'inicialització és només el valor inicial utilitzat per iniciar un procés iteratiu.
- La majoria dels sistemes criptogràfics simètrics requereixen un IV aleatori i imprevisible, o almenys únic per a cada missatge encriptat amb una clau determinada.
- Aquest IV aleatori garanteix que cada missatge s'encripta de manera diferent tot i utilitzar la mateixa clau, de manera que veure múltiples missatges codificats amb la mateixa clau no proporciona informació addicional.
- El IV, a més, assegura que encriptar el mateix missatge dues vegades produeix dos xifrats completament diferents.
- En qualsevol cas, el IV mai ha de mantenir-se secret. De fet, en la majoria dels casos, mantenir en secret el IV no seria pràctic, ja que el destinatari no podria desxifrar les dades.

AES amb C#

- **AES** es pot implentar a .NET utilitzant la classe **AesManaged** que proporciona una implementació administrada de l'algoritme simètric Advanced Encryption Standard (AES).
- També caldrà implementar la Interfície **ICryptoTransform** que defineix les operacions bàsiques de les transformacions criptogràfiques
- Per a les dades caldrà utilitzar un **MemoryStream** que asseguri que la informació no es guarda en cap moment en el disc i un **CryptoStream** que defineix un stream que vincula el flux de dades amb les transformacions criptogràfiques.

La classe AesManaged

- Les propietats més rellevants són:

IV	Obté o estableix el vector d'inicialització (IV) que s'utilitzarà per a l'algoritme simètric.
Key	Obté o estableix la clau secreta utilitzada per a l'algoritme simètric
KeySize	Obté o estableix la mida, en bits, de la clau secreta utilitzada per a l'algoritme simètric.

- Els camps més rellevants són:

IVValue	Representa el vector d'inicialització (IV) per a l'algoritme simètric.
KeySizeValue	Representa la mida, en bits, de la clau secreta utilitzada
KeyValue	Representa la clau secreta per a l'algoritme simètric.

La classe AesManaged

- Els mètodes més rellevants són:

Clear	Allibera tots els recursos utilitzats per la classe
CreateDecryptor	Crea un objecte de desxifrat simètric amb la clau actual i el vector d'inicialització (IV).
CreateEncryptor	Crea un objecte de xifrat simètric amb la clau actual i el vector d'inicialització (IV).
GenerateIV	Genera un vector d'inicialització aleatori (IV) que s'utilitza per a l'algoritme simètric.
GenerateKey	Genera una clau aleatòria per utilitzar per a l'algoritme simètric.
ValidKeySize	Determina si la mida de la clau especificada és vàlida per a l'algoritme actual.

Exemple AesManaged

```
string original = “Aquesta és la dada a encriptar S2AM!!!";

using (AesManaged myAes = new AesManaged())
{
 myAes.Key = Key;
 myAes.IV = IV;
 byte[] encrypted;

 ICryptoTransform encryptor = myAes.CreateEncryptor(myAes.Key, myAes.IV);
 using (MemoryStream msEncrypt = new MemoryStream())
  {
   using (CryptoStream csEncrypt =
          new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))
    {
     using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
      {
       swEncrypt.Write(plainText);
      }
     encrypted = msEncrypt.ToArray();
    }
  }
```

CryptoStream

- Utilitzar **CryptoStream** en C# és bastant senzill. En primer lloc, cal disposar d'un stream base que s'utilitzarà com a memòria intermèdia per al xifrat/desxifrat.
- També es necessitarà d'un transformador criptogràfic que formi part de la classe **CryptographicServiceProvider**.
- La combinació d'aquestes parts permet xifrar/desxifrar en runtime.

```
FileStream stream = new FileStream("C:\\test.txt",
                    FileMode.OpenOrCreate,FileAccess.Write);

AESCryptoServiceProvider cryptic = new AESCryptoServiceProvider();
cryptic.Key = ASCIIEncoding.ASCII.GetBytes("ABCDEFGH");
cryptic.IV = ASCIIEncoding.ASCII.GetBytes("ABCDEFGH");
CryptoStream crStream = new CryptoStream(stream,
cryptic.CreateEncryptor(),CryptoStreamMode.Write);
byte[] data = ASCIIEncoding.ASCII.GetBytes("Hola S2AM!!!");
crStream.Write(data,0,data.Length);
crStream.Close();
stream.Close();
```

Bibliografia

- C# AES 256 bits Encryption Library with Salt
- AesManaged Class
- Using CryptoStream in C#
- AES encryption in C#
- How to encrypt and decrypt data using a symmetric key
- How to securely handle AES “Key” and “IV” values
- CrossAES C# Example
- Where to store a server side encryption key?
- Best way to store encryption keys in .NET C#

Criptografia asimètrica

- Un sistema de criptografia asimètrica o de clau pública utilitza claus diferents per encriptar i desencriptar.
- Per a realitzar encriptació amb clau simètrica el sistema criptogràfic més utilitzat és **RSA**.
- **RSA** (Rivest-Shamir-Adleman) és un dels primers criptosistemes de clau pública i és àmpliament utilitzat per a la transmissió segura de dades. En aquest criptosistema, la clau de xifratge és pública i és diferent de la clau de desxifrat que es manté en secret (privada). A RSA, aquesta asimetria es basa en la dificultat pràctica de la factorització del producte de dos nombres primers grans.
- RSA és un algorisme relativament lent, i per això, s'utilitza poc per xifrar directament les dades dels usuaris.
- L'ús més habitual de RSA és encriptar les claus simètriques de tipus AES es faran servir per a realitzar operacions de xifrat-desxifrat a gran velocitat.

RSA amb C#

- Per tal d'utilitzar RSA, .NET proporciona la classe **RSA** de la qual derivarà la classe **RSACryptoServiceProvider** .
- Aquesta classe proporciona mètodes que permeten realitzar una sèrie d'operacions necessàries per a la gestió d'un sistema criptogràfic de clau pública.
  - Crear les claus publica i privada i poder gestionar-les.
  - És necessari poder guardar la clau privada de forma segura sense guardar-la com a text pla en el disc de l'equip
  - És necessari poder exportar la clau pública i poder distribuir-la entre els usuaris que hagin de xifrar missatges que haguem de rebre.
  - Caldrà poder encriptar utilitzant la clau pública i desencriptar utilitzant la clau privada.

Creació i gestió de claus

- La classe **RSACryptoServiceProvider** crea un parell de claus pública/privada quan s'utilitza el constructor per defecte. Aquestes claus asimètriques es poden emmagatzemar per utilitzar-les en sessions múltiples o poden ser generades només per a una sessió.
- La informació de les claus es pot extreure mitjançant un dels següents mètodes:
  - El mètode **ExportParameters**
  - El mètode **ToXmlString**
- El mètode **ExportParameters**, que retorna una estructura **RSAParameters** que conté la informació de la clau.

```
//Això genera un parell de claus pública\privada.
RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();

//Salvem la info de les claus a una estructura RSAParameters.
RSAParameters RSAKeyInfo = RSA.ExportParameters(false);
```

Creació i gestió de claus

- El mètode **ToXmlString**, que retorna una representació XML de la informació de la clau.

```
//Això genera un parell de claus pública\privada.
RSACryptoServiceProvider rsa = new RSACryptoServiceProvider();

//Salvem la info de les claus en un XML.
string publicKey = rsa.ToXmlString(false);
string privateKey = rsa.ToXmlString(true);

File.WriteAllText("c:\\temp\\PublicKey.xml", publicKey);
```

- Ambdós mètodes accepten un valor booleà que indica si només es retornarà la informació de la clau pública o es retornarà la informació de la clau pública i de la clau privada.
- Està totalment desaconsellat persistir la clau privada.

Creació i gestió de claus

- Exemple de clau privada XML

```
<RSAKeyValue>
        <Modulus>qxLgukh ( ... ) Zw15KwgEfMlIkc=</Modulus>
        <Exponent>AQAB</Exponent>
        <P>zNliyTEj ( ... ) DQlOAVw==</P>
        <Q>1cp1efO ( ... ) K5bd3kQ==</Q>
        <DP>ySrGJz ( ... ) VFmdH0mw==</DP>
        <DQ>gxRNDuaZs ( ... ) As5sekYQ==</DQ>
        <InverseQ>g8Jxm ( ... ) 9uvxCZOyaA==</InverseQ>
        <D>K4uwnHpboi8 ( ... ) 5KNMU8E=</D>
</RSAKeyValue>
```

- Exemple de clau pública XML

```
<RSAKeyValue>
        <Modulus>qxLgukh ( ... ) Zw15KwgEfMlIkc=</Modulus>
        <Exponent>AQAB</Exponent>
</RSAKeyValue>
```

Creació i gestió de claus

- Les claus privades asimètriques mai no s'han d'emmagatzemar en forma de text pla a l'equip local. Si cal emmagatzemar una clau privada, s'haurà d'utilitzar un contenidor de claus o **KeyContainer .**
- Per a dur a terme aquesta operació cal:
  - Crear un objecte **CspParameters** i passar el nom que es vol donar al contenidor de claus al camp **KeyContainerName**.
  - Tot seguit es crea un nou objecte **RSACryptoServiceProvider** passant-li l'objecte **CspParameters** creat prèviament com a paràmetre al seu constructor.

```
CspParameters cspp = new CspParameters();
const string keyName = "Key01";

cspp.KeyContainerName = keyName;
RSACryptoServiceProvider rsa = new RSACryptoServiceProvider(cspp);
rsa.PersistKeyInCsp = true;
```

Creació i gestió de claus

- Opcionalment es pot utilitzar la propietat **PersistKeyInCsp** que indica si la volem guardar (fer persistent o no). Per defecte el seu valor és **true** si s'inicialitza l'objecte **RSACryptoServiceProvider** passant un objecte **CspParameters** al seu constructor.
- Per a poder reutilitzar les claus emmagatzemades, podem inicialitzar una classe **RSACryptoServiceProvider** passant-li el valor d'una estructura **RSAParameters** mitjançant el mètode **ImportParameters**.

```
using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())
{
 RSAParameters rsaParams = rsa.ExportParameters(false);
 //Creem un altre objecte RSACryptoServiceProvider.
 using (RSACryptoServiceProvider rsa2 = new RSACryptoServiceProvider())
  {
   //Importem la info del primer rsa al segon
   rsa2.ImportParameters(RSAParams);
  }
 }
```

Creació i gestió de claus

- Per poder distribuir la clau pública el millor mètode és utilitzar XML. Aquest XML es pot emmagatzemar en una BBDD per tal que els usuaris autoritzats la puguin obtenir.
- Per poder carregar una clau pública des d'un fitxer XML cal utilitzar el mètode **FromXmlString** que permet inicialitzar un objecte RSA amb la clau carregada des de un fitxer XML.

```
rsaEnc = new RSACryptoServiceProvider();
string xmlKey = File.ReadAllText("c:\\temp\\PublicKey.xml");
rsaEnc.FromXmlString(xmlKey);
```

La classe RSACryptoServiceProvider

- Les propietats més rellevants són:

CspKeyContainerInfo	Obté un objecte **CspKeyContainerInfo** que descriu informació addicional sobre el parell de claus criptogràfiques.
KeySize	Obté la mida de la clau actual.
PersistKeyInCsp	Obté o estableix un valor que indica si la clau ha de ser persistida en el proveïdor de serveis criptogràfics .
PublicOnly	Obté un valor que indica si l'objecte **RSACryptoServiceProvider** conté només una clau pública
UseMachineKeyStore	Obté o estableix un valor que indica si la clau hauria de continuar al contenidor de claus de l'ordinador en lloc del contenidorde claus del perfil d'usuari.

- Els camps més rellevants són:

KeySizeValue	Representa la mida, en bits, del mòdul de la clau utilitzat per l'algorisme asimètric.

La classe RSACryptoServiceProvider

- Els mètodes més rellevants són:

Clear	Allibera tots els recursos utilitzats per la classe **RSACryptoServiceProvider**
Decrypt	Desxifra les dades amb l'algoritme **RSA**
Encrypt	Xifra les dades amb l'algoritme **RSA**.
ExportParameters	Exporta els **RSAParameters**.
FromXmlString	Inicialitza un objecte RSA amb una clau carregada des d'un fitxer XML.
ImportParameters	Importa els **RSAParameters**.
ToXmlString	Crea i retorna una cadena XML que conté la clau de l'objecte **RSA** actual

Exemple RSA

```
static void Main()
{
 UnicodeEncoding ByteConverter = new UnicodeEncoding();
 byte[] dataToEncrypt = ByteConverter.GetBytes("Hola S2AM!!!");
 byte[] encryptedData;
 byte[] decryptedData;

 using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider())
 {
  encryptedData =
    RSAEncrypt(dataToEncrypt, RSA.ExportParameters(false), false);
  decryptedData =
    RSADecrypt(encryptedData, RSA.ExportParameters(true), false);
  Console.WriteLine("Text: {0}", ByteConverter.GetString(decryptedData));
 }
}
```

Exemple RSA

```
public static byte[] RSAEncrypt(byte[] DataToEncrypt,
                RSAParameters RSAKeyInfo, bool DoOAEPPadding)
{
  byte[] encryptedData;
  using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider())
  {
   RSA.ImportParameters(RSAKeyInfo);
   encryptedData = RSA.Encrypt(DataToEncrypt, DoOAEPPadding);
  }
   return encryptedData;
}

public static byte[] RSADecrypt(byte[] DataToDecrypt,
                RSAParameters RSAKeyInfo, bool DoOAEPPadding)
{
 byte[] decryptedData;
 using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider())
 {
  RSA.ImportParameters(RSAKeyInfo);
  decryptedData = RSA.Decrypt(DataToDecrypt, DoOAEPPadding);
 }
 return decryptedData;
    }
```

Bibliografia

- Encryption (part 2, basics, advanced), RSA
- RSACryptoServiceProvider Class
- How to Generate Public/Private Key Using RSA
- Generate Public\Private Keys in C# and RSA
- RSA Algorithm With C#
- Walkthrough: Creating a Cryptographic Application

Encriptació de XML

- El namespace **System.Security.Cryptography.Xml** permet xifrar elements en un document XML.
- Es pot utilitzar el xifratge XML per reemplaçar qualsevol element o document XML amb un element **<EncryptedData>** que conté les dades xifrades del XML. L'element **<EncryptedData>** també pot contenir sub elements que inclouen informació sobre les claus i els processos utilitzats durant el xifratge.
- En un escenari habitual s'encripta un element XML mitjançant dues claus.
  - Es generen un parell de claus **RSA** pública/privada i es guarden de forma segura.
  - A continuació es crea una clau de sessió simètrica mitjançant l'algoritme **AES** i es xifren els elements XML desitjats.
  - Després s'utilitza la clau pública **RSA** per xifrar la clau de sessió **AES**.
  - Finalment es desa la clau de sessió AES xifrada i les dades xifrades XML al document XML en un nou element **<EncryptedData>.**

Encriptació de XML

- Assumint que tenim les claus RSA emmagatzemades (bé en un **KeyContainer** si s'ha d'encriptar/desencriptar en un mateix equip o bé en format XML en BBDD si cal encriptar en un equip i desencriptar en un altre) (Veure RSA)
- El següent pas a fer és carregar el document en un objecte **XmlDocument** i especificar quin element volem encriptar

```
XmlDocument xmlDoc = new XmlDocument();

// Carrega el document especificat a l'objecte XmlDocument.
xmlDoc.PreserveWhitespace = true;
xmlDoc.Load("Dades.xml");

// Obté l'element especificat que caldrà xifrar.
String ElementToEncrypt = "DadesSecretes"
XmlElement elementToEncrypt =
Doc.GetElementsByTagName(ElementToEncrypt)[0] as XmlElement;
```

Encriptació de XML

- Es crea una nova clau de sessió mitjançant una classe simètrica (en aquest cas usant **RijndaelManaged**).

```
// Es crea una clau de sessió de 256 bits
sessionKey = new RijndaelManaged();
sessionKey.KeySize = 256;
```

- Es genera una nova instància de la classe **EncryptedXml** i s'utilitza el seu mètode **EncryptData** per xifrar l'element especificat anteriorment mitjançant la clau simètrica de sessió.
- El mètode **EncryptData** retorna l'element xifrat com una matriu de bytes xifrats.

```
EncryptedXml eXml = new EncryptedXml();
byte[] encryptedElement =
       eXml.EncryptData(elementToEncrypt, sessionKey, false);
```

Encriptació de XML

- S'instancia un objecte **EncryptedData** i es configura amb l'identificador d'URL de l'element XML xifrat i el seu ID.
- El camp **XmlEncElementUrl** permet especificar l'identificador d'URL fàcilment.
- L'element XML original en text pla serà reemplaçat per un element **<EncryptedData>** encapsulat per aquest objecte **EncryptedData**.

```
EncryptedData edElement = new EncryptedData();
edElement.Type = EncryptedXml.XmlEncElementUrl;
edElement.Id = EncryptionElementID;
```

- Es crea un objecte **EncryptionMethod** especificant el identificador URL utilitzat pel agorisme criptogràfic.

```
edElement.EncryptionMethod =
          new EncryptionMethod(EncryptedXml.XmlEncAES256Url);
```

Encriptació de XML

- Cal crear un objecte **EncryptedKey** per contenir la clau de sessió xifrada. Es xifra la clau de sessió mitjançant el mètode **EncryptedXml.EncryptKey**
- El mètode **EncryptedXml.EncryptKey** s'inicialitza passant-li la clau de sessió i l'objecte **RSACryptoServiceProvider** que caldrà utilitzar per xifrar la clau i retorna la clau de sessió xifrada en forma d'array de bytes.

```
EncryptedKey ek = new EncryptedKey();
byte[] encryptedKey = EncryptedXml.EncryptKey(sessionKey.Key, RSA, false);
```

- Es crea un objecte **CipherData** que representarà l'element que contindrà les dades encriptades, i s'instancia un objecte **EncryptionMethod** que encapsularà l'algorisme utilitzat per encriptar el XML

```
ek.CipherData = new CipherData(encryptedKey);
ek.EncryptionMethod = new EncryptionMethod(EncryptedXml.XmlEncRSA15Url);
```

Encriptació de XML

- De forma opcional es pot crear un nou objecte **DataReference** que relacioni les dades xifrades amb una clau de sessió en particular. Això permet especificar fàcilment que diverses parts d'un document XML estan xifrades per una única clau.

```
DataReference dRef = new DataReference();
dRef.Uri = "#" + EncryptionElementID;
ek.AddReference(dRef);
```

- Cal afegir la clau xifrada a l'objecte **EncryptedData**.

```
edElement.KeyInfo.AddClause(new KeyInfoEncryptedKey(ek));
```

- Es crea un objecte **CipherData** que representarà l'element que contindrà les dades encriptades, i s'instancia un objecte **EncryptionMethod** que encapsularà l'algorisme utilitzat per encriptar el XML

Encriptació de XML

- Es crea un nou objecte **KeyInfo** per especificar el nom de la clau RSA i s'afegeix a l'objecte **EncryptedData**. D'aquesta manera, la part que desxifra identificarà la clau asimètrica correcta que s'utilitzarà al desxifrar la clau de sessió.

```
KeyInfoName kin = new KeyInfoName();
kin.Value = KeyName;
ek.KeyInfo.AddClause(kin);
```

- Cal afegir les dades de l'element xifrat a l'objecte **EncryptedData**.

```
edElement.CipherData.CipherValue = encryptedElement;
```

- Finalment es substitueix l'element original de l'objecte **XmlDocument** amb l'element que té l'objecte **EncryptedData** i es desa l'objecte **XmlDocument**.

```
EncryptedXml.ReplaceElement(elementToEncrypt, edElement, false);
xmlDoc.Save("DadesEnc.xml");
```

Desencriptació de XML

- El namespace **System.Security.Cryptography.Xml** permet desxifrar elements en un document XML.
- Es pot utilitzar el desxifrat XML per trobar un element **<EncryptedData>,** desxifra-lo i, a continuació, substituir-lo l'element XML original en text pla.
- En un escenari habitual es desencripta un element XML mitjançant dues claus.
  - Es recupera la clau privada RSA generada prèviament des d'un **KeyContainer**
  - S'utilitza la **clau RSA** per desxifrar una **clau de sessió AES** emmagatzemada en l'element **<EncryptedKey>** de l'element **<EncryptedData>.**
  - A continuació s'utilitza la clau de sessió AES per desxifrar l'element XML.

Desencriptació de XML

- Assumint que tenim les claus RSA emmagatzemades en un **KeyContainer,** es crea un objecte **CspParameters** amb el nom del **KeyContainer** que guarda la clau privada.
- S'instancia un nou **RSACryptoServiceProvider** configurat amb el **CspParameters** de manera que la clau que conté es recupera per a ser utilitzada.

```
CspParameters cspParams = new CspParameters();
cspParams.KeyContainerName = "KeyPlanet";
RSACryptoServiceProvider rsaKey = new RSACryptoServiceProvider(cspParams);
```

- Cal crear un objecte EncryptedXml nou per desxifrar el document.

```
EncryptedXml exml = new EncryptedXml(Doc);
```

Desencriptació de XML

- S'Afegeix una assignació de clau/nom per associar la clau RSA amb l'element del document que s'ha de desxifrar. Cal utilitzar el mateix nom per a la clau que es va utilitzar quan va xifrar el document.

```
exml.AddKeyNameMapping(KeyName, Alg);
```

- Es crida al mètode **DecryptDocument** per desxifrar l'element **<EncryptedData>.** Aquest mètode utilitza la clau RSA per desxifrar la clau de sessió AES i utilitzar-la per desxifrar l'element XML. També substitueix automàticament l'element **<EncryptedData>** amb el text pla original.

```
exml.DecryptDocument();
```

- Es desa el document XML ja desxifrat

```
xmlDoc.Save("DadesDes.xml");
```

Signatura digital de XML

- El namespace **System.Security.Cryptography.Xml** permet signar un document XML o una part d'un document XML amb una signatura digital.
- Les signatures digitals XML permeten verificar que les dades d'un XML no s'han modificat després de la signatura.
- En un escenari habitual es signa un element XML mitjançant criptografia de clau pública (asimètrica).
  - Es crea una clau de signatura RSA, i es guarda de forma segura
  - Aquesta clau s'utilitza per signar digitalment el document XML.

Signatura digital de XML

- Es crea un objecte **CspParameters** especificant un nom del **KeyContainer.**
- Es genereu una clau asimètrica amb la classe **RSACryptoServiceProvider**. La clau es guarda automàticament al contenidor especificat si es passa l'objecte **CspParameters** al constructor de la classe.

```
CspParameters cspParams = new CspParameters();
cspParams.KeyContainerName = "KeyPlanet";
RSACryptoServiceProvider rsaKey = new RSACryptoServiceProvider(cspParams);
```

- Es carrega el document a signar en un **XmlDocument**.

```
XmlDocument xmlDoc = new XmlDocument();
xmlDoc.PreserveWhitespace = true;
xmlDoc.Load("DocOriginal.xml");
```

Signatura digital de XML

- Es crea un nou objecte SignedXml al qual es passa l'objecte XmlDocument.

```
SignedXml signedXml = new SignedXml(xmlDoc);
```

- S'afegeix la clau RSA de signatura a l'objecte SignedXml. Es crea un objecte **Reference** per especificar què es vol signar. En cas de voler signar el document sencer, cal establir la propietat **Uri** en "".

```
signedXml.SigningKey = key;
Reference reference = new Reference();
reference.Uri = "";
```

- Cal afegir un objecte **XmlDsigEnvelopedSignatureTransform** a l'objecte Referència. Aquesta classe elimina l'element **<Signature>** d'un document XML abans de calcular el resum. Amb aquesta transformació, es pot verificar el document en la seva versió original.

Signatura digital de XML

- Amb la transformació, **XmlDsigEnvelopedSignatureTransform** es pot signar i verificar tots els elements d'un document XML, excepte els elements de signatura digital XML.

```
signedXml.AddReference(reference);
```

- Es calcula la signatura mitjançant el mètode **ComputeSignature**.

```
signedXml.ComputeSignature();
```

- Es recupera l'element **<Signature>,** es guarda en un nou objecte **XmlElement** i s'afegeix al **XmlDocument**. Es desa el document XML.

```
XmlElement xmlDigitalSignature = signedXml.GetXml();
xmlDoc.DocumentElement.
            .AppendChild(xmlDoc.ImportNode(xmlDigitalSignature, true));
xmlDoc.Save("test.xml");
```

Verificar la Signatura de XML

- El namespace **System.Security.Cryptography.Xml** permet verificar les dades XML signades amb una signatura digital..
- Les signatures digitals XML permeten verificar que les dades d'un XML no s'han modificat després de la signatura.
- Per verificar el document, s'ha d'utilitzar la mateixa clau asimètrica que es va utilitzar per signar que s'ha de recuperar del **KeyContainer** o de la BBDD en format XML. En aquest exemple s'utilitzarà una clau carregada des de un KeyContainer mitjançant un objecte CspParameters que es passa al constructor de l'objecte RSACryptoServiceProvider

```
CspParameters cspParams = new CspParameters();
cspParams.KeyContainerName = "KeyPlanet";
RSACryptoServiceProvider rsaKey = new RSACryptoServiceProvider(cspParams);
```

Verificar la Signatura de XML

- Es carrega el document a verificar en un **XmlDocument**.

```
XmlDocument xmlDoc = new XmlDocument();
xmlDoc.PreserveWhitespace = true;
xmlDoc.Load("DocOriginal.xml");
```

- Es crea un nou objecte **SignedXml** al qual es passa l'objecte **XmlDocument**.

```
SignedXml signedXml = new SignedXml(xmlDoc);
```

- Es busca l' element **<Signature>** i es crea un nou objecte **XmlNodeList**.

```
XmlNodeList nodeList = Doc.GetElementsByTagName("Signature");
```

Verificar la Signatura de XML

- Es carrega el XML del primer element **<Signature>** en l'objecte **SignedXml**.

```
signedXml.LoadXml((XmlElement)nodeList[0]);
```

- Es comprova la signatura mitjançant el mètode **CheckSignature** i la clau pública RSA. Aquest mètode retorna un valor booleà que indica l'èxit o el fracàs de la verificació.

```
return signedXml.CheckSignature(Key);
```

Bibliografia

- How to: Encrypt XML Elements with Asymmetric Keys
- How to: Decrypt XML Elements with Asymmetric Keys
- How to: Sign XML Documents with Digital Signatures
- How to: Verify the Digital Signatures of XML Documents