BẢNG THEO DÕI SỬA ĐỔI

STT	Version	Nội dung sửa đổi	Ngày có hiệu lực
1	v1.0	Viết mới	11/07/2014

MỤC LỤC

1. TỔNG QUAN VỀ CHỮ KÝ SỐ

1.1. Chứng thư số

- Chứng thư số là một dạng chứng thư điện tử do tổ chức cung cấp dịch vụ chứng thực chữ ký số cấp.
- Chứng thư số có thể được xem như là một "chứng minh thư" sử dụng trong môi trường máy tính và Internet.
- Chứng thư số được sử dụng để nhận diện một cá nhân, một máy chủ, hoặc một vài đối tượng khác và gắn định danh của đối tượng đó với một khóa công khai (public key), được cấp bởi những tổ chức có thẩm quyền xác nhận định danh và cấp các chứng thư số.

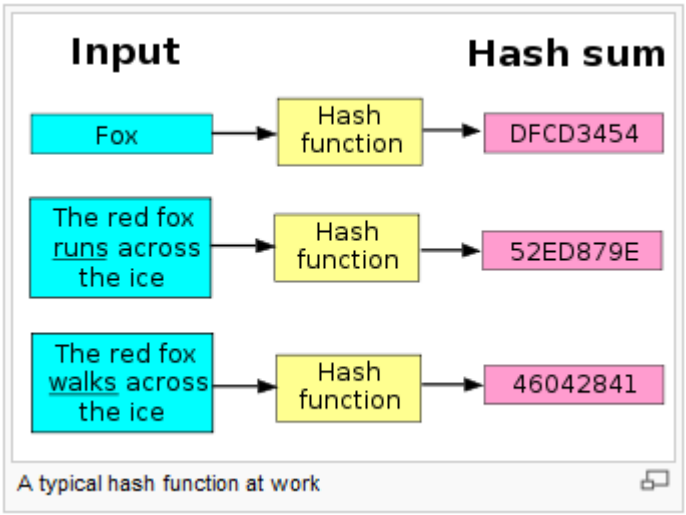
1.2. Chữ ký số

- Chữ ký điện tử: là thông tin (âm thanh, ký hiệu, tiến trình điện tử…) đi kèm theo dữ liệu (văn bản, hình ảnh, video…) nhằm mục đích xác định chủ của dữ liệu đó.
- Chữ ký số: là tập con của chữ ký điện tử. Chữ ký số là chữ ký điện tử dựa trên kỹ thuật mã hóa với khóa công khai và khóa bí mật.

1.3. Hàm băm

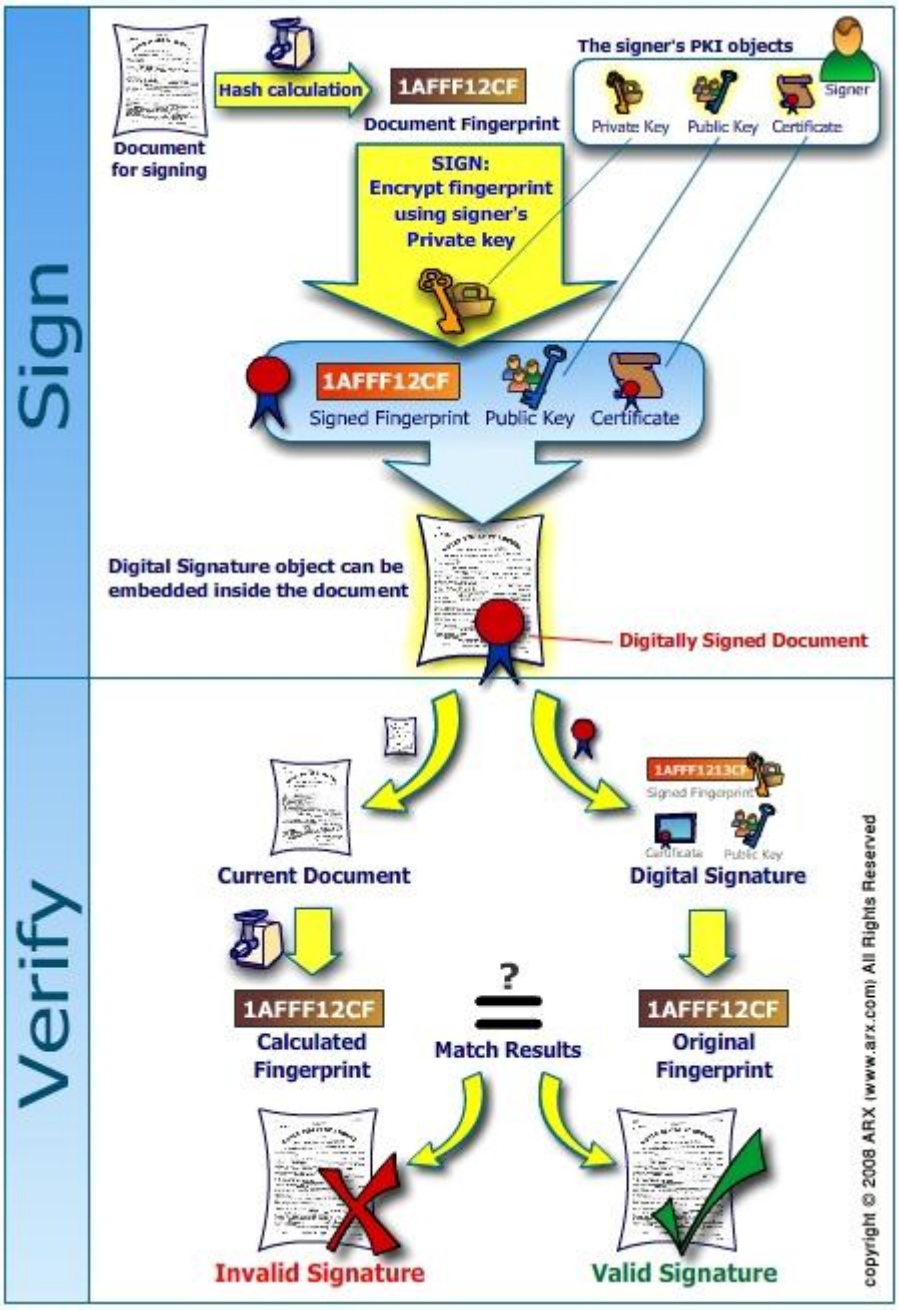
Các giải thuật băm (hash) ánh xạ các giá trị nhị phân có độ dài tùy ý thành những giá trị nhị phân nhỏ hơn có một độ dài quy định trước, kết quả được gọi là giá trị băm. Một giá trị băm đại diện cho dữ liệu duy nhất. Với các hàm băm được sử dụng trong ký số là các hàm băm mạnh. Nghĩa là:

- Với 2 dữ liệu gần giống nhau sẽ cho 2 giá trị băm khác nhau.
- Không thể suy ngược từ giá trị băm ra dữ liệu gốc.



A typical hash function at work

1.4. Mô hình ký số



1.4.1. Ký số

Quy trình ký số:

- Bước 1: Dữ liệu được đưa qua hàm băm A thu giá trị băm. Ví dụ: 1AFFF12CF
- Bước 2: Mã hóa giá trị băm này bằng khóa bí mật được giá trị YYYYYYYYY.
- Bước 3: Kết hợp giá trị băm đã được mã hóa này (YYYYYYYYY) với khóa công khai để được chữ ký số.
- Bước 4: Đính kèm chữ ký số này vào dữ liệu ban đầu được dữ liệu đã ký.

1.4.2. Xác thực chữ ký số

Quy trình xác thực chữ ký số:

- Bước 1: Dữ liệu có chứa chữ ký số được tác ra làm 2 phần: dữ liệu gốc và chữ ký số.

- Bước 2: Cho dữ liệu gốc đi qua hàm băm A (đã sử dụng trong quá trình ký số ở trên) để thu được giá trị băm: XXXXXXXXX
- Bước 3: Dùng khóa công khai để giải mã YYYYYYYYY để được giá trị băm 1AFFF12CF.
- Bước 4: So sánh giá trị XXXXXXXXX với 1AFFF12CF. Nếu bằng nhau thì ta có thể kết luận được:

- Dữ liệu là dữ liệu được ký số và chưa bị thay đổi từ lúc ký.
- Người ký lên văn bản chính là người sử hữu chứng thư số tương ứng với khóa công khai.

1.5. Tính chất của chữ ký số

Sử dụng chữ ký số đảm bảo:
- Tính xác thực nguồn gốc: kiểm tra được định danh người ký lên dữ liệu, chống giả mạo.
- Tính toàn vẹn dữ liệu: kiểm tra được dữ liệu có bị thay đổi hay không.
- Tính chống chối bỏ.

1.6. Lợi ích

Sử dụng chữ ký số mang lại những hiệu quả thiết thực:
- Nâng cao hiệu quả trong công tác quản lý, sản xuất kinh doanh. Xử lý công việc mọi lúc, mọi nơi.
- Tiết kiệm chi phí.
- Giải phóng sức lao động.
- Đơn giản hóa quy trình nghiệp vụ.

2. TÍCH HỢP CHỮ KÝ SỐ

2.1. Đăng ký

Khi áp dụng chứng thư số, mỗi người dùng đăng ký một chứng thư số với ứng dụng. Ứng dụng thực hiện lưu chứng thư số phục vụ chức năng xác thực chữ ký số.

2.2. Ký số (sign)

2.2.1. Ứng dụng ký trên hệ thống client-server

Với các ứng dụng áp dụng chữ ký trên hệ thống client server, ví dụ client là thiết bị của người dùng, lưu khóa bí mật, sẽ thực hiện chức năng ký, server gửi về client nội dung cần ký. Client thực hiện ký và gửi về thông tin chữ ký cho server. Server đính chữ ký vào dữ liệu. Các bước thực hiện:
- Server thực hiện tạo giá trị băm tương ứng với dữ liệu cần ký, tạo trường lưu trữ chữ ký số trên file và lưu lại thành file tạm:
  createDigest(srcFile, tempFile, certChain);
- Server gửi giá trị băm về cho client (cùng với các thông tin cần thiết khác để xác định dữ liệu ký)
- Client thực hiện ký lên giá trị băm (mã hóa sử dụng khóa bí mật):
  sign(hash, privateKey);
- Client gửi lại chuỗi đã ký cho server.
- Server đính kèm chữ ký số này vào dữ liệu:
  insertSignature(tempFile, destFile, signature);

2.2.2. Ứng dụng ký trực tiếp

Với các ứng dụng ký trực tiếp lên dữ liệu, khóa bí mật và khóa công khai (chứng thư số) phải được lưu trữ ngay trên cơ sở dữ liệu của ứng dụng. Ứng dụng thực hiện các bước ký như quy trình mặc định (tạo hash, ký số, đính kèm chữ ký vào dữ liệu)

3. EXAMPLE CODE

3.1. PDF

3.1.1. Ký số

- Khởi tạo:

```
BouncyCastleProvider provider = new BouncyCastleProvider();
Security.addProvider(provider);
KeyStore ks = KeyStore.getInstance("pkcs12");
ks.load(new FileInputStream(KEYSTORE), PASSWORD);
String alias = (String) ks.aliases().nextElement();
PrivateKey pk = (PrivateKey) ks.getKey(alias, PASSWORD);
Certificate[] chain = ks.getCertificateChain(alias);
```

- Ký:

```
// Creating the reader and the stamper
PdfReader reader = new PdfReader(src);
FileOutputStream os = new FileOutputStream(dest);
PdfStamper stamper = PdfStamper.createSignature(reader, os, '\0');

// Creating the appearance
PdfSignatureAppearance appearance = stamper.getSignatureAppearance();
appearance.setReason(reason);
appearance.setLocation(location);
appearance.setVisibleSignature(new Rectangle(36, 48, 144, 80), 1, "sig");

// Creating the signature
ExternalDigest digest = new BouncyCastleDigest();
ExternalSignature signature =
   new PrivateKeySignature(pk, digestAlgorithm, provider);
```

- Đính kèm chữ ký số:

```
byte[] paddedSig = new byte[estimatedSize];
System.arraycopy(encodedSig, 0, paddedSig, 0, encodedSig.length);
PdfDictionary dic2 = new PdfDictionary();
dic2.put(PdfName.CONTENTS, new PdfString(paddedSig).setHexWriting(true));
```

3.1.2. Xác thực chữ ký số

- Khởi tạo:

```
Security.addProvider(new org.bouncycastle.jce.provider.BouncyCastleProvider());
PdfReader pdfReader = new PdfReader(SIGNED_FILE);
AcroFields af = pdfReader.getAcroFields();
```

- Lấy trường chữ ký số và xác thực:

```
ArrayList<String> names = af.getSignatureNames();
for (String name : names) {
   PdfPKCS7 pk = af.verifySignature(name);
   Certificate[] pkc = pk.getCertificates();
   System.out.println("verify: " + pk.verify());
}
```

3.2. Excel (xls, xlsx)

3.2.1. Ký số

- Khởi tạo:

```
KeyStore ks = KeyStore.getInstance("pkcs12");
ks.load(new FileInputStream(KEYSTORE), PASSWORD);
String alias = (String) ks.aliases().nextElement();
PrivateKey pk = (PrivateKey) ks.getKey(alias, PASSWORD);
Certificate[] chain = ks.getCertificateChain(alias);
XLSXDigitalSignature ods = new XLSXDigitalSignature(pk, chain);
XLSXDigitalSignature.initial();
```

- Tạo giá trị băm:

```
byte[] hash = ods.createHash(FILE_EXCEL);
```

- Tạo chữ ký:

```
byte[] signatureValue = ods.signOOXMLDigest(hash);
```

- Insert chữ ký vào file:

```
ods.insertSignature(signatureValue, FILE_EXCEL_OUT);
```

3.2.2. Xác thực chữ ký số

- Khởi tạo:

```
OoxmlSignatureVerifier verifier = new OoxmlSignatureVerifier();
verifier.initial();;
KeyStore ks = KeyStore.getInstance("pkcs12");
ks.load(new FileInputStream(KEYSTORE), PASSWORD);
```

- Xác thực:

```
Verifier v = verifier.verify((new File(FILE_EXCEL_OUT).toURI().toURL()), ks, null).get(0);
System.out.println("Valid ?: " + v.isValid());
System.out.println("Modified ? : " + v.isModified());
```

3.3. XML

3.3.1. Ký số

- Khởi tạo:

```
KeyStore ks = KeyStore.getInstance("pkcs12");
ks.load(new FileInputStream(KEYSTORE), PASSWORD);
String alias = (String) ks.aliases().nextElement();
PrivateKey pk = (PrivateKey) ks.getKey(alias, PASSWORD);
Certificate[] chain = ks.getCertificateChain(alias);
File temp = File.createTempFile("temp-file-name", ".tmp");
```

- Tạo giá trị băm:

```
DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
dbFactory.setNamespaceAware(true);
Document doc = dbFactory.newDocumentBuilder().parse(src);

// prepare signature factory
String providerName =
System.getProperty("jsr105Provider","org.jcp.xml.dsig.internal.dom.XMLDSigRI");
final XMLSignatureFactory sigFactory = XMLSignatureFactory.getInstance("DOM",
(Provider) Class.forName(providerName).newInstance());
```

```
doc.getElementsByTagName("HSoKhaiThue").item(0);
Node sigParent = doc.getDocumentElement();
String referenceURI = ""; // Empty string means whole document
List transforms = Collections.singletonList(sigFactory.newTransform(
Transform.ENVELOPED, (TransformParameterSpec) null));

// Create a Reference to the enveloped document
Reference ref = sigFactory.newReference(referenceURI,
sigFactory.newDigestMethod(DigestMethod.SHA1, null), transforms, null, null);
// Create the SignedInfo
SignedInfo signedInfo = sigFactory.newSignedInfo(
sigFactory.newCanonicalizationMethod(CanonicalizationMethod.INCLUSIVE,
(C14NMethodParameterSpec) null), sigFactory.newSignatureMethod(
SignatureMethod.RSA_SHA1, null), Collections.singletonList(ref));

// Create the SignedInfo.
X509Certificate cert = (X509Certificate) chain[0];
KeyInfoFactory keyInfoFactory = sigFactory.getKeyInfoFactory();
List x509Content = new ArrayList();
 x509Content.add(cert);
x509Content.add(chain[1]);
X509Data xd = keyInfoFactory.newX509Data(x509Content);
KeyInfo keyInfo = keyInfoFactory.newKeyInfo(Collections.singletonList(xd));

// Create a DOMSignContext and specify the RSA PrivateKey and
 // location of the resulting XMLSignature's parent element
KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");
KeyPair kp = kpg.generateKeyPair();
DOMSignContext dsc = new DOMSignContext(kp.getPrivate(), sigParent);

// Create the XMLSignature (but don't sign it yet)
XMLSignature signature = sigFactory.newXMLSignature(signedInfo, keyInfo);
// Marshal, generate (and sign) the enveloped signature
signature.sign(dsc);
byte[] hash =
IOUtils.toByteArray(signature.getSignedInfo().getCanonicalizedData());
//Save temp file
Transformer trans = TransformerFactory.newInstance().newTransformer();
StreamResult res = new StreamResult(new FileOutputStream(tempFile));
trans.transform(new DOMSource(doc), res);
```

- Tạo chữ ký:

```
Signature sig = Signature.getInstance("SHA1withRSA");
sig.initSign(privateKey);
sig.update(hash);
byte[] signature = sig.sign();
```

- Insert chữ ký vào dữ liệu:

```
DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
dbFactory.setNamespaceAware(true);
Document doc = dbFactory.newDocumentBuilder().parse(tempFile);
String signatureXML = Base64.encode(extSignature);
Node signatureValue = doc.getElementsByTagName("SignatureValue").item(0);
```

```
signatureValue.setTextContent(signatureXML);
Transformer trans = TransformerFactory.newInstance().newTransformer();
StreamResult res = new StreamResult(new FileOutputStream(dest));
trans.transform(new DOMSource(doc), res);
```

3.3.2. Xác thực chữ ký số

- Khởi tạo:

```
// Instantiate the document to be validated
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
dbf.setNamespaceAware(true);
Document doc = dbf.newDocumentBuilder().parse(new
FileInputStream(SIGNED_FILE));
```

- Tách lấy chữ ký:

```
// Find Signature element
NodeList nl = doc.getElementsByTagNameNS(XMLSignature.XMLNS, "Signature");
if (nl.getLength() == 0) {
        throw new Exception("Cannot find Signature element");
}

// Create a DOM XMLSignatureFactory that will be used to unmarshal the
// document containing the XMLSignature
XMLSignatureFactory fac = XMLSignatureFactory.getInstance("DOM");

// Create a DOMValidateContext and specify a KeyValue KeySelector
// and document context
KeyInfoKeySelector keySelector = new KeyInfoKeySelector();
DOMValidateContext valContext = new DOMValidateContext(keySelector,
nl.item(0));
// unmarshal the XMLSignature
XMLSignature signature = fac.unmarshalXMLSignature(valContext);
```

- Xác thực:

```
// Validate the XMLSignature (generated above)
boolean coreValidity = signature.validate(valContext);
X509Certificate cert = keySelector.getCertificate();
Certificate[] chain = keySelector.getCertificateChain();

// Check core validation status
if (coreValidity == false) {
        System.err.println("Signature failed core validation");
        boolean sv = signature.getSignatureValue().validate(valContext);
        System.out.println("signature validation status: " + sv);
        // check the validation status of each Reference
        Iterator i = signature.getSignedInfo().getReferences().iterator();
        for (int j = 0; i.hasNext(); j++) {
                boolean refValid = ((Reference) i.next()).validate(valContext);
                System.out.println("ref[" + j + "] validity status: " + refValid);
         }
} else {
        System.out.println("Signature passed core validation");
}
```

3.3.3. *Xác thực chữ ký số*

- Khởi tạo:

```
OoxmlSignatureVerifier verifier = new OoxmlSignatureVerifier();
verifier.initial();;
KeyStore ks = KeyStore.getInstance("pkcs12");
ks.load(new FileInputStream(KEYSTORE), PASSWORD);
```

- Xác thực:

```
Verifier v = verifier.verify((new File(FILE_EXCEL_OUT).toURI().toURL()), ks, null).get(0);
System.out.println("Valid ?: " + v.isValid());
System.out.println("Modified ? : " + v.isModified());
```