FIRMA ELECTRÓNICA DE LA FACTURA ELECTRÓNICA ECUATORIANA

31 agosto, 2016

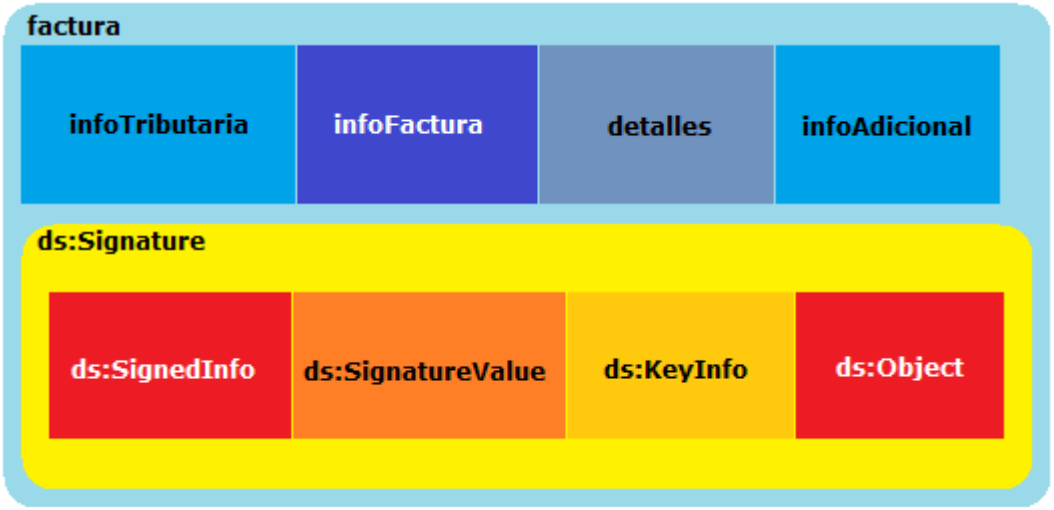
En el Ecuador el uso de facturas electrónicas (y en general de comprobantes electrónicos) es obligatorio a cada vez más tipos de empresas. Su implementación requiere que tengan firma electrónica ecuatoriana en estándar XAdES-BES (acrónimo de *XML Advanced Electronic Signatures – Basic Electronic Signature*), sin embargo no especifican con suficiente detalle la forma de implementar tal formato de firma, viéndose las empresas obligadas a utilizar un applet de Java que realice la acción como caja negra. A nivel técnico, esto quería decir que, no importa el lenguaje de programación utilizado, sea Python , .Net, PHP, u otro, era necesario salir de ese entorno para llamar a la funcionalidad en Java, con las molestias y bajas de rendimiento del caso.

A continuación se explicará la implementación de la firma electrónica XAdES BES que acepta el Servicio de Rentas Internas del Ecuador -SRI-. Como prerrequisito se necesita el certificado digital .P12 que lo vende el Banco Central del Ecuador. También es útil la herramienta gratuita del SRI para la facturación electrónica, con la cual se pueden comparar los resultados del firmado.

También te puede interesar la parte dos del artículo donde se implementa la firma digital en JavaScript y se explica el proceso de ingeniería inversa con el cual se obtuvo estos resultados. Adicional, en la parte tres del artículo se listan algunos problemas comunes que surgen en las implementaciones de este algoritmo, se detallan sus causas y se presentan posibles soluciones.

Estructura de la firma electrónica XAdES BES

La estructura de una factura electrónica con firma electrónica en formato XAdES BES válida es la siguiente:



Estructura de la factura electrónica con su firma electrónica
El nodo `ds:Signature` contiene toda la firma generada y tiene dos *namespaces*:

1. `xmlns:ds="http://www.w3.org/2000/09/xmldsig#"`
2. `xmlns:etsi="http://uri.etsi.org/01903/v1.3.2#"`

A continuación se detallan las cuatro partes de la firma electrónica XAdES.

`ds:SignedInfo`

Guarda el método de canonicalización (c14n), el algoritmo de firmado (SHA1) y tres *hash* que referencian a los siguientes nodos:

1. `SignedProperties` dentro del nodo `ds:Object` (detallado más abajo en este artículo), que tiene los datos adicionales para implementar el formato XAdES BES.
2. `ds:KeyInfo`, con la información del certificado firmante.
3. `comprobante`, con la información de la factura electrónica.

Para obtener estos *hash* es necesario aplicar tanto el método de canonicalización como el algoritmo de firmado previamente mencionados. En la práctica, la canonicalización consiste solamente en agregar los dos *namespaces* antes indicados al nodo que se va a firmar, respetando todos los saltos de línea, espacios y caracteres especiales . Por ejemplo:

Original:

```
<ds:KeyInfo Id="Certificate976903">

    ...

</ds:KeyInfo>
```

Para obtener el *hash*:

```
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmln
s:etsi="http://uri.etsi.org/01903/v1.3.2#" Id="Certificate9769
03">

    ...

</ds:KeyInfo>
```

La excepción sería el hash del nodo `comprobante`, el cual no se canonicaliza previamente.

Como se mencionó, el algoritmo utilizado para el *hash* es SHA1, el cual por cierto está siendo reemplazado a nivel mundial por SHA256 debido a las vulnerabilidades descubiertas en su algoritmo. Esperemos que el SRI mejore la seguridad en este sentido.

Para su inclusión en el archivo XML de la factura electrónica firmada, el *hash* debe ser codificado en base64.

`ds:SignatureValue`

Es la firma digital codificada en base64 del nodo `ds:SignedInfo`, previamente canonicalizado.

Actualización al 3 de enero del 2018: Esta parte de la firma, aunque parece relativamente simple (solo se obtendría la firma digital del nodo indicado, no es así?), puede llegar a ser un poco problemática, tal y como se lo detalla en la parte tres del artículo.

`ds:KeyInfo`

Guarda la información del certificado digital con el que se realiza la firma electrónica.

Tiene dos nodos: `ds:X509Data` y `ds:KeyValue`.

- El nodo `ds:X509Data` tiene un solo nodo hijo, `ds:X509Certificate`, el cual guarda el certificado en formato PEM (acrónimo de *Privacy Enhanced Mail*, que es la versión codificada en base64 del certificado digital y que originalmente se creó para enviar los certificados binarios vía correo electrónico, de ahí su nombre), sin las cabeceras de inicio y final, y en renglones de 76 caracteres.
- El nodo `ds:KeyValue` tiene un nodo hijo, `ds:RSAKeyValue`, y dos nodos nietos, `ds:Modulus` y `ds:Exponent`. El primero guarda el módulo en base64, y el segundo guarda el exponente, el cual es 65537 codificado en base64, que resulta `AQAB`. Por cierto, el número 65537 es un número primo de Fermat, exactamente el F4, que es usado ampliamente en algoritmos de encriptación por su bajo peso de Hamming, lo cual es eficiente en cálculos realizados por máquinas binarias.

`ds:Object`

Almacena información adicional para implementar el estándar XAdES BES. Sin esta sección no serían firmas XAdES sino firmas electrónicas XML clásicas.

Este nodo tiene un solo nodo hijo, `etsi:QualifyingProperties`, y un solo nodo nieto, `etsi:SignedProperties`. Este último tiene dos nodos hijos, `etsi:SignedSignatureProperties` y `etsi:SignedDataObjectProperties`.

- El nodo `etsi:SignedSignatureProperties` tiene dos nodos hijos, `etsi:SigningTime` y `etsi:SigningCertificate`. El primero guarda

fecha y hora de generación en formato ISO 8601 **en hora local** (YYYY-MM-DDThh:mm:ssTZD, donde TZD es la zona horaria en formato «-hh:mm»), y el segundo tiene la siguiente estructura:

- `etsi:Cert`
    - `etsi:CertDigest`
        - `ds:DigestMethod`**:** El mismo algoritmo SHA1 usado anteriormente.
        - `ds:DigestValue`**:** el hash del certificado **en formato DER** (*Distinguished Encoding Rules*), codificado en base64.
    - `etsi:IssuerSerial`
        - `ds:X509IssuerName`**:** guarda el valor constante (por ahora) de `"CN=AC BANCO CENTRAL DEL ECUADOR,L=QUITO,OU=ENTIDAD DE CERTIFICACION DE INFORMACION-ECIBCE,O=BANCO CENTRAL DEL ECUADOR,C=EC"`. Sería recomendable sacar este valor del mismo certificado digital, para mantener compatibilidad futura.
        - `ds:X509SerialNumber`**:** el número serial del certificado digital, en formato decimal, no hexadecimal.
- El nodo `etsi:SignedDataObjectProperties` tiene la siguiente estructura:
    - `etsi:DataObjectFormat`
        - `etsi:Description`: guarda el valor constante de `"contenido comprobante"`.
        - `etsi:MimeType`: guarda el valor constante de `"text/xml"`.

Números aleatorios en la estructura de la firma electrónica

Existen ocho números aleatorios enteros entre 1 y 100 mil, que se colocan a lo largo de la estructura de la firma electrónica:

1. `Certificate_number`
2. `Signature_number`
3. `SignedProperties_number`
4. `SignedInfo_number`
5. `SignedPropertiesID_number`
6. `Reference_ID_number`
7. `SignatureValue_number`
8. `Object_number`

Estos números aleatorios se colocan en atributos de varios nodos, de la siguiente forma:

1. En `ds:KeyInfo`, su atributo `Id` se compone de la siguiente manera: `"Certificate" & Certificate_number`.
2. En el segundo nodo hijo `ds:Reference` del nodo `ds:SignedInfo`, el atributo `URI` se compone de la siguiente manera: `"#Certificate" & Certificate_number`.
3. En `etsi:SignedProperties`, su atributo `Id` se compone de la siguiente manera: `"Signature" & Signature_number & "-SignedProperties" & SignedProperties_number`.
4. En el primer nodo hijo `ds:Reference` del nodo `ds:SignedInfo`, el atributo `URI` se compone de la siguiente manera: `"#Signature" & Signature_number & "-SignedProperties" & SignedProperties_number`.

5. En `ds:Signature`, su atributo `Id` se compone de la siguiente manera: `"Signature" & Signature_number`.
6. En `ds:Object`, su atributo `Id` se compone de la siguiente manera: `"Signature" & Signature_number & "-Object" & Object_number`.
7. En `etsi:QualifyingProperties`, su atributo `Target` se compone de la siguiente manera: `"#Signature" & Signature_number`.
8. En `ds:SignedInfo`, su atributo `Id` se compone de la siguiente manera: `"Signature-SignedInfo" & SignedInfo_number`.
9. En el primer nodo hijo `ds:Reference` del nodo `ds:SignedInfo`, el atributo `Id` se compone de la siguiente manera: `"SignedPropertiesID" & SignedPropertiesID_number`.
10. En `etsi:DataObjectFormat`, su atributo `ObjectReference` se compone de la siguiente manera: `"#Reference-ID-" & Reference_ID_number`.
11. En el tercer nodo hijo `ds:Reference` del nodo `ds:SignedInfo`, el atributo `Id` se compone de la siguiente manera: `"Reference-ID-" & Reference_ID_number`.
12. En `ds:SignatureValue`, su atributo `Id` se compone de la siguiente manera: `"SignatureValue" & SignatureValue_number`.

Implementación en JavaScript

Para la implementación de la firma electrónica en JavaScript se utilizó la librería Forge tanto para la obtención de los *hash* SHA1 así como para la extracción de datos del certificado digital, y el firmado en sí. También se utilizó la librería moment.js para la obtención de la fecha y hora de generación.

Para probar la firma electrónica se tiene dos opciones:

1. Firmar primero con el facturador electrónico gratuito del SRI, luego colocar los ocho números aleatorios presentes en la firma

electrónica generada, y con esos números ejecutar la versión JavaScript del generador de firmas. Deben resultar en los mismos valores.
2. Generar la firma electrónica con la versión JavaScript, codificar todo el contenido del XML resultante en base64, y enviar el texto vía SOAP a los Servicios Web del SRI.

El código fuente se lo puede acceder en GitHub.

Actualización al 25 de octubre de 2016:

Se creó una prueba de concepto donde la firma electrónica se genera 100% local en el navegador, sin recursos de servidor ni enviando información a ninguna parte. Se la puede acceder en https://jsfiddle.net/jybaro/xmvov1c3/

Esta versión, a más de utilizar las librerías Forge y Moment mencionadas antes, también llama a la librería buffer, la cual permite manipular información binaria en el navegador.