SHCP
SECRETARÍA DE HACIENDA Y CRÉDITO PÚBLICO

SAT
Servicio de Administración Tributaria

Lineamientos técnicos para que las personas autorizadas proporcionen su sello digital en los pedimentos que prevaliden

CONTROL DE VERSIONES

VERSION	FECHA	ELABORÓ	COMENTARIOS
1.0	08-12-2016	Administración Central de Modernización Aduanera.	

INDICE

I. Objetivo.

Dar a conocer a las personas que cuenten con autorización para realizar la prevalidación de los pedimentos en términos del artículo 16-A de la Ley Aduanera las especificaciones tecnológicas que deben de cumplir para proporcionar su sello digital en los pedimentos que prevaliden.

II. Marco Jurídico

Ley Aduanera.
Artículo 16-A.

Código Fiscal de la Federación.

Reglamento de la Ley Aduanera
Artículo 13.

Reglamento Interior del Servicio de Administración Tributaria.

Reglas Generales de Comercio Exterior vigente.
Regla 1.82., fracción III.

Lineamientos Técnicos de Registros VOCE-SAAI M3 Versión 8.0

III.LINEAMIENTOS

Primero. Las personas que cuenten con autorización para realizar la prevalidación de los pedimentos deberán proporcionar su sello digital en los pedimentos que prevaliden.

Una vez que el prevalidador verifique que el archivo de validación del pedimento cumpla con los criterios sintácticos, catalógicos, estructurales y normativos establecidos, deberán insertar su sello digital.

El sello digital se podrá obtener, a través del software CERTIFICA (antes SOLCEDI) que se encuentra disponible en la página de internet www.sat.gob.mx en el apartado de e.firma.

Una vez descargado el software, se solicitará el certificado de sello digital utilizando la e.firma vigente del prevalidador y en el campo de "Nombre de la sucursal o unidad" se deberá colocar la palabra "PREVALIDADOR" en mayúsculas.

Realizada la solicitud de sello digital, se descargará el certificado del sello digital de la página electrónica del SAT, en la sección correspondiente.

El archivo de validación del pedimento con la constante .m (constante con la que se identifican los archivos de pedimentos SAAI M3) puede incluir información de uno o más pedimentos, esto conforme a los Lineamientos Técnicos de Registros VOCE-SAAI M3 Versión 8.0, por lo que el sello digital del archivo corresponderá a cada uno de los pedimentos contenidos en éste.

Segundo: Un pedimento con sello digital de la persona que realice la prevalidación tiene las siguientes características:

- Es infalsificable.
- El sello digital de un archivo de validación no es reciclable (es único por archivo).
- Una cadena original de un archivo de validación del pedimento sellado digitalmente que hubiese sido alterada, es detectable.
- Una cadena original de un archivo de validación del pedimento sellada digitalmente no puede ser repudiada.

Tercero: Se deberá utilizar el algoritmo de digestión SHA2-256 para realizar la digestión del contenido del archivo de validación del pedimento.

El SHA2-256, es una función hash (digestión, picadillo o resumen) de un solo sentido tal que para cualquier entrada produce una salida compleja de 160 bits (20 bytes) denominada "digestión".

Cuarto. Para el Sellado del archivo de Pedimentos se empleará el estándar **CMS** (de sus siglas en inglés *Cryptographic Message Syntax*, Sintaxis de Mensajes Criptográficos) de la **IETF** (por sus siglas en inglés, *Internet Engineering Task Force*[1]) documentada en el **RFC 5652**[2].

Quinto. El procedimiento para generar el sello es el siguiente[3]:

a) Leer el *contenido* del archivo de validación del pedimento o pedimentos a sellar.

b) Realizar la *digestión* del *contenido* empleando el algoritmo de digestión SHA2-256.

c) Firmar digitalmente la digestión obtenida en el paso anterior empleando la *llave privada* del prevalidador autorizado

d) Colectar el valor de la *firma digital* y del *certificado* (CDS) del *firmante* en la *información del firmante* (*SignerInfo*).

e) Generar los *datos firmados* (*SignedData*) incorporando la *información del firmante* (*SignerInfo*) y el *contenido*.

f) Grabar los *datos firmados* (*SignedData*) en el archivo de salida que será transmitido al SAT para su validación.

Para la implementación del procedimiento citado pueden emplear diversas librerías de uso libre, dependiendo de la plataforma de desarrollo, como son BouncyCastle[4] y OpenSSL[5].

[1] The Internet Engineering Task Force (IETF). https://www.ietf.org/
[2] Cryptographic Message Syntax (CMS). https://tools.ietf.org/html/rfc5652
[3] Signed-Data Content Type. https://tools.ietf.org/html/rfc5652#page-8
[4] The Legion of BouncyCastle. http://www.bouncycastle.org
[5] OpenSSL. https://www.openssl.org

Ejemplo

A continuación, se muestra código JAVA de ejemplo como referencia.

```
/** Generar archivo firmado con CMS. */
private void firmarArchivo(CMSTypedData data, String nombreArchivoCert,
        String privateKeyLocation)
                    throws IOException,
                    CertificateEncodingException,
                    OperatorCreationException,
                    CMSException {

      // Leer archivo de Certificado para Sellos Digitales (CSD)
      FileInputStream fisCert = new FileInputStream(nombreArchivoCert);
      CertificateFactory cf = CertificateFactory.getInstance("X.509");
      X509Certificate cert =
      (X509Certificate)cf.generateCertificate(fisCert);

      // Agregamos el CSD a la lista de certificados
      ArrayList<X509Certificate> certList = new
      ArrayList<X509Certificate>();
      certList.add(publicKey);
      JcaCertStore certs = new JcaCertStore(certList);

      //Leer llave privada
      RandomAccessFile raf = new RandomAccessFile(privateKeyLocation,
      "r");
      byte[] buf = new byte[(int)raf.length()];
      raf.readFully(buf);
      raf.close();

      PKCS8EncodedKeySpec kspec = new PKCS8EncodedKeySpec(buf);
      KeyFactory kf = KeyFactory.getInstance("RSA");
      PrivateKey privateKey = kf.generatePrivate(kspec);

      CMSSignedDataGenerator gen = new CMSSignedDataGenerator();
      ContentSigner SHA256Signer = new JcaContentSignerBuilder(
                                        gen.DIGEST_SHA256)
                                        .build(privateKey);

      gen.addSignerInfoGenerator(
              new JcaSignerInfoGeneratorBuilder(
                  new JcaDigestCalculatorProviderBuilder()
                  .build())
              .setDirectSignature(true)
              .build(sha256Signer, publicKey));

      // Se añade el certificado a la lista
      gen.addCertificates(certs);

      // Se genera la firma
      CMSSignedData sigData = gen.generate(data, true);
```

```
        // Se escribe el resultado en el archivo de salida
        FileOutputStream fosSalida = new
        FileOutputStream(nombreArchivoFirmado);
        fosSalida.write(sigData.getEncoded());
        fosSalida.close();
}
```