Iberpay

Always one step ahead

Seguridad en los Servicios WEB

Manual Técnico

Versión 1.2

Junio de 2021

[Uso restringido]

ÍNDICE

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

0. INFORMACIÓN DEL DOCUMENTO

0.1 Relación de versiones

Versiones	Estado	Fecha
Manual Técnico sobre la seguridad en los servicios web v1.0	Borrador	Enero de 2021
Manual Técnico sobre la seguridad en los servicios web v1.1	Borrador	Marzo 2021
Manual Técnico sobre la seguridad en los servicios web v1.2	Versión en vigor	Junio 2021

0.2 Objetivo del documento

Este manual describe los aspectos técnicos de seguridad relacionados con la firma y cifrado de mensajes de los servicios web de Iberpay, y que igualmente deben cumplirse por parte de las entidades con conexión técnica directa a estos servicios de Iberpay.

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

1. SEGURIDAD EN LOS SERVICIOS WEB CON MENSAJES SOAP

1.1 Introducción

El protocolo SOAP (Simple Object Access Protocol) es un protocolo ligero basado en XML para el intercambio de información entre entornos descentralizados, distribuidos, permitiendo la interacción entre varios dispositivos con la capacidad de transmitir información compleja. Los mensajes que se transmiten son independientes del sistema operativo y de los protocolos.

El mensaje SOAP está formado por un envelope, que es el elemento principal del mensaje y su estructura está compuesta por header y body:

- El header es el elemento principal del envelope, se trata un mecanismo de extensión que incorpora información extra en el mensaje SOAP (como seguridad, transacciones, etc.).
- El body especifica una solicitud para efectuar alguna operación, un resultado o error.
- El elemento fault que se encuentra dentro del cuerpo que indica un error en el procesamiento del mensaje SOAP, y está formado de cinco subelementos: code, reason, detail, node y role.

Un mensaje SOAP es una vía de transmisión de información desde un emisor a un receptor. Este mensaje, puede pasar a través de varios intermediarios que, a su vez, pueden tratar el mensaje. El conjunto de intermediarios por donde viaja el mensaje se conoce como ruta del mensaje. Todos los usuarios que intervienen en la ruta por la que viaja el mensaje se conocen como actores.

Debido a que SOAP está basado en XML, es vulnerable a una gran cantidad de ataques orientados al mismo. Además, también presenta vulnerabilidades a los ataques asociados con su protocolo de capa de aplicación, con mayor frecuencia HTTP.

Como SOAP es utilizado para servicios web sus vulnerabilidades se clasifican de la siguiente forma:

- Vulnerabilidades estructurales: son aquellas que derivan de la intervención en las líneas de comunicación de los participantes (usuarios, equipos intermedios, servidor) que intercambian mensajes SOAP. Dan lugar a los siguientes ataques: Interceptación de mensajes, Man-in-the middle, Repetición de mensajes, Denegación de Servicio, etc.
- Vulnerabilidades semánticas: son aquellas que se derivan de las debilidades en la codificación y procesamiento de los mensajes SOAP. Dan lugar a los siguientes ataques: entradas inválidas, control de acceso débil, autenticación y manejo de sesión débiles, cross

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

site scripting, buffer overflows, injection flaws, manejo inapropiado de errores, contenido malicioso, denegación de servicio.

1.2 Método de seguridad en los mensajes SOAP

Para garantizar la seguridad de los mensajes SOAP Iberpay utiliza el estándar **WS-Security**. Dicho estándar define especificaciones que implementan una serie de mejoras en el tratamiento de mensajería SOAP con el objetivo de mejorar la protección de los mensajes basándose en dos mecanismos esenciales: la integridad y confidencialidad de los mensajes; y la autenticación de un mensaje de forma individual.

WS-Security es flexible y su diseño constituye la base para la creación de modelos de seguridad más complejos incluyendo PKI, Kerberos y SSL. En particular, WS-Security proporciona soporte para múltiples tokens de seguridad, múltiples dominios de confianza, múltiples formatos de firma y múltiples tecnologías de cifrado. Dentro de las múltiples políticas que admite WS-Security, los servicios de Iberpay utilizan la opción **Token Ws-Security 1.0 con certificado X509**.

Para el empleo de esta política, se usarán juegos de claves públicas y privadas, en función de si se trata de exponer el servicio, o de consumirlo, al emplear un algoritmo de clave asimétrica para su firma. Por ello, Iberpay, deberá compartir su clave pública con las entidades bancarias y estas, a su vez, deberán compartir con Iberpay su clave pública (este aspecto se describe en la sección 3).

Token WS Security-1.0 X509 refuerza la protección a nivel de mensaje (integridad y confidencialidad) y el llenado de credenciales de certificado/autenticación mediante los mecanismos descritos en WS-Security 1.0:

- Algorithm Suite: son conjuntos de estándares de diferentes combinaciones de algoritmos, que traen predefinidas una serie de mecanismos de seguridad.
- Digest: es una propiedad de resumen, que especifica el algoritmo utilizado para generar un valor de resumen de mensaje.
- Encryption: es una propiedad de cifrado, que especifica el algoritmo utilizado para cifrar los datos.
- Symmetric Key Wrap: es una propiedad que define el encapsulado de clave simétrica.
- Asymmetric Key Wrap: es una propiedad que define el encapsulado de clave asimétrica.
- Encrypted Key Derivation: es una propiedad que define el algoritmo empleado para la derivación de las claves que se van a usar para la encriptación.

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

- Signature Key Derivation: es una propiedad que define el algoritmo empleado para la derivación de las claves que se van a usar para la firma.
- Minimum Signature Key Length: es una propiedad que define el tamaño mínimo de la firma del mensaje.
- Symmetric Signature: esta propiedad de firma de clave simétrica especifica el algoritmo para generar una firma utilizando una clave simétrica.
- Asymmetric Signature: esta propiedad de firma de clave asimétrica especifica el algoritmo para generar una firma utilizando una clave asimétrica.

La política que usa Iberpay es **WS-Security's Basic 256 con tecnologías de claves asimétricas**, concretamente:

Algorithm Suite	Basic256Sha256
Digest	Sha-256
Encryption	Aes256
Symmetric Key Wrap	KwAes256
Asymmetric Key Wrap	KwRsaOaep
Encrypted Key Derivation	PSha1L256
Signature Key Derivation	PSha1L192
Minimum Signature Key Length	256
Symmetric Signature	HmacSha1
Asymmetric Signature	RsaSha1

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

1.3 Mecanismo de firma y cifrado del mensaje SOAP

Los pasos generales para crear y cifrar mensajes SOAP conformes con la especificación WS-Security se muestran a continuación:

1. Crear un nuevo sobre SOAP.
2. Crear una cabecera de seguridad wsse:Security.
3. Localizar los elementos de datos que van a ser cifrados.
4. Cifrar los elementos de datos de la siguiente manera: Para cada elemento XML o contenido del elemento dentro del sobre SOAP destino, cifrarlo de acuerdo con las reglas de procesamiento definidas en la especificación XML Encryption. Cada elemento o contenido del elemento original seleccionado debe ser eliminado y reemplazado por el elemento resultante xenc:EncryptedData.

5. El elemento ds:KeyInfo dentro del elemento xenc:EncryptedKey podría referenciar a otro elemento ds:KeyInfo. Destacar que si el cifrado está basado en un token de seguridad anexado, entonces el elemento SecurityTokenReference deberá ser añadido al elemento ds:KeyInfo para facilitar su localización.
6. Crear un elemento xenc:DataReference que referencie los elementos xenc:EncryptedData generados. Añadir el elemento creado xenc:DataReference al elemento xenc:Reference-List incluido dentro de la cabecera de seguridad.
7. Firmar los elementos Body y Timestamp.
8. Encriptar el contenido del Body.
9. Enviar el mensaje.

1.4 Mecanismo de validación de firma y descifrado del mensaje SOAP

Cuando se recibe un mensaje SOAP con entradas de cabeceras cifradas, para cada cabecera cifrada se deben seguir los siguientes pasos generales para su procesado:

1. Localizar los elementos xenc:EncryptedData que deben ser descifrados.
2. Descifrar como se indica a continuación: para cada elemento en el sobre SOAP destino, descifrarlo de acuerdo con las reglas de procesamiento descritas en la especificación XML Encryption y las reglas de procesamiento ya descritas con anterioridad en este mismo documento.

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

1.5 Ejemplo de mensajes

Ejemplos de request y response SOAP:

```
?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://spf.iberpay.es/exposed/WS_ENVIO_OPER">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
      <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" wsu:Id=xxx</wsse:BinarySecurityToken>
      <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="EK-9CC88C34BCA290CFCF1624357256708617">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p" />
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#9CC88C34BCA290CFCF1624357256708618" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>xxxx</xenc:CipherValue>
        </xenc:CipherData>
        <xenc:ReferenceList>
          <xenc:DataReference URI="#ED-9CC88C34BCA290CFCF1624357256708619" />
        </xenc:ReferenceList>
      </xenc:EncryptedKey>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="SIG-9CC88C34BCA290CFCF1624357256701616">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
            <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="soapenv ws" />
          </ds:CanonicalizationMethod>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
          <ds:Reference URI="#id-9CC88C34BCA290CFCF1624357256701615">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="ws" />
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
            <ds:DigestValue>gcaMqVrVnE+Oq5b7Z0mt8tNN2eg=</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>xxx</ds:SignatureValue>
        <ds:KeyInfo Id="KI-9CC88C34BCA290CFCF1624357256701613">
          <wsse:SecurityTokenReference wsu:Id="STR-9CC88C34BCA290CFCF1624357256701614">
            <wsse:KeyIdentifier EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3">xxx</wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
      <wsu:Timestamp wsu:Id="TS-9CC88C34BCA290CFCF1624357256699611">
        <wsu:Created>2021-06-22T10:20:56.699Z</wsu:Created>
        <wsu:Expires>2021-06-22T10:25:56.699Z</wsu:Expires>
      </wsu:Timestamp>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="id-9CC88C34BCA290CFCF1624357256701615">
    <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="ED-9CC88C34BCA290CFCF1624357256708619" Type="http://www.w3.org/2001/04/xmlenc#Content">
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsse11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#EncryptedKey">
          <wsse:Reference URI="#EK-9CC88C34BCA290CFCF1624357256708617" />
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>xxx</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
```

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
      <xenc:EncryptedKey Id="EK-aa298539-a740-489c-a3fa-ad85345d4f54" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
          <wsse:SecurityTokenReference>
            <ds:X509Data>
              <ds:X509IssuerSerial>
                <ds:X509IssuerName>CN=miguel canovas,OU=iberpay,O=iberpay,L=Madrid,ST=Madrid,C=ES</ds:X509IssuerName>
                <ds:X509SerialNumber>27427347</ds:X509SerialNumber>
              </ds:X509IssuerSerial>
            </ds:X509Data>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:xxx>
        </xenc:CipherData>
        <xenc:ReferenceList>
          <xenc:DataReference URI="#ED-202c17bc-436d-4b5f-a815-ff44c9227a9f"/>
        </xenc:ReferenceList>
      </xenc:EncryptedKey>
      <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
wsu:Id="X509-6a91cb7b-26e5-4b3c-816b-xxx</wsse:BinarySecurityToken>
      <ds:Signature Id="SIG-6072ac27-aa52-4168-a5fa-9ece34f43f75" xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
            <ec:InclusiveNamespaces PrefixList="soap" xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          </ds:CanonicalizationMethod>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          <ds:Reference URI="#TS-5caf09cc-7e1a-4ef5-a766-c388bb0284d9">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <ec:InclusiveNamespaces PrefixList="soap wsse" xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>AxyOtc96rwDIbCwJ41R58iDwdtQ=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#id-70c5efcf-f21d-449b-a576-c22e70196e20">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>/rq4lOFbI4f0dlhAGjrpySMbe6g=</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>xxx</ds:SignatureValue>
        <ds:KeyInfo Id="KI-1222a1d1-5177-43f3-a2f4-45fc126af551">
          <wsse:SecurityTokenReference wsu:Id="STR-141c4bf6-af9e-44ec-a8da-c38758b9bf29">
            <wsse:Reference URI="#X509-6a91cb7b-26e5-4b3c-816b-54ca7c65fe95" ValueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
      <wsu:Timestamp wsu:Id="TS-5caf09cc-7e1a-4ef5-a766-c388bb0284d9">
        <wsu:Created>2021-06-22T10:58:26.021Z</wsu:Created>
        <wsu:Expires>2021-06-22T11:03:26.021Z</wsu:Expires>
      </wsu:Timestamp>
    </wsse:Security>
  </soap:Header>
  <soap:Body wsu:Id="id-70c5efcf-f21d-449b-a576-c22e70196e20" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
    <xenc:EncryptedData Id="ED-202c17bc-436d-4b5f-a815-ff44c9227a9f" Type="http://www.w3.org/2001/04/xmlenc#Content"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <wsse:SecurityTokenReference wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#En-
cryptedKey" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsse11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd">
          <wsse:Reference URI="#EK-aa298539-a740-489c-a3fa-ad85345d4f54"/>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>xxx</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </soap:Body>
</soap:Envelope>
```

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

2. SEGURIDAD EN LOS SERVICIOS WEB CON MENSAJES REST

2.1 Introducción

El protocolo REST (REpresentatinal State Transfer) es un protocolo ligero empleado para implementar servicios web ligeros y de fácil consumo entre cliente y servidor. El modo de comunicación REST es muy sencillo, debido a que varios clientes que se encuentren enganchados a la red, de los que no importa la naturaleza pero que deben soportar el protocolo HTTP, pueden llevar a cabo peticiones.

Las principales vulnerabilidades que existen en los servicios que utilizan mensajería tipo REST son:

- Vulnerabilidades de la información: los atacantes explotan los atributos enviados en una API como en URL, el cuerpo de la petición/respuesta o en las cabeceras HTTP.
- Vulnerabilidades en autenticación/autorización: tratan la identificación de la aplicación, cuya identificación está asociada a unas credenciales y en muchos casos a una API Key. Si esta se almacena en el propio código, sería relativamente sencilla de usurpar.
- Vulnerabilidades en transporte: con APIs que no estén configuradas correctamente empleando SSL/TLS

2.2 Método de seguridad en los mensajes REST

JSON Web Token, es un estándar (RFC 7519) abierto basado en JSON y propuesto por IETF para la creación de tokens de acceso, que permiten la propagación de identidad y privilegios. Dicha información puede ser verificada y confiable al estar firmada digitalmente.

Para la securización de servicios REST con JSON Iberpay utiliza **JWE** (JSON Web Encryption), que estandariza la forma de representar un contenido cifrado en una estructura de datos basada en JSON. Existen múltiples posibilidades, por lo que las operaciones criptográficas aplicadas en JWE se describen a través de encabezados JOSE (Javascript Object Signing and Encryption).

Para garantizar la confidencialidad, integridad y autenticidad, Iberpay utiliza en sus servicios WEB el cifrado autenticado, utilizando pares de claves públicas/privadas.

En concreto, Iberpay utiliza el algoritmo **RSA_OAEP-256** con el método de cifrado **AES256-GCM**, que se define en la especificación JWA. Este utiliza el estándar de cifrado avanzado (AES) en el algoritmo Galois/Counter Mode (GCM) con una clave larga de 256 bits, y es un algoritmo de clave simétrica utilizado para AEAD. Las claves simétricas se utilizan principalmente para el cifrado de contenido y es mucho más rápido que el cifrado de claves asimétricas.

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

2.3 Mecanismo de cifrado de la carga útil JSON

Los pasos para cifrar el JSON de un servicio con mensajería tipo REST son:

- El elemento enc del encabezado JOSE define el algoritmo de cifrado de contenido y debe ser un algoritmo simétrico de cifrado autenticado con datos asociados (AEAD).
- El elemento alg del encabezado JOSE define el algoritmo de cifrado para cifrar la clave de cifrado de contenido (CEK).
- El elemento kid identifica la clave pública que se ha utilizado para el cifrado del token.
- El cifrado autenticado con datos asociados (AEAD) es un modo de operación de cifrado en bloque que proporciona simultáneamente garantías de confidencialidad, integridad y autenticidad de los datos; el descifrado se combina en un solo paso con la verificación de la integridad.
- Para el cifrado de contenido, se utiliza el algoritmo A256GCM; y para envoltura de claves, RSA-OAEP.
- Con RSA-OAEP, durante el proceso de cifrado, el emisor del token genera una clave aleatoria, que tiene un tamaño de 256 bits y cifra el mensaje utilizando esa clave siguiendo el algoritmo AES GCM. A continuación, la clave utilizada para cifrar el mensaje se cifra mediante RSA-OAEP, que es un esquema de cifrado asimétrico. El esquema de cifrado RSA-OAEP utiliza el algoritmo RSA con el método Optimal Asymmetric Encryption Padding (OAEP). Finalmente, la clave simétrica cifrada se coloca en la sección Encabezado cifrado JWE de JWE.
- El cuarto elemento del token JWE es el valor codificado en base64url del texto cifrado JWE. El texto cifrado JWE se calcula cifrando la carga útil JSON de texto sin formato utilizando la clave de cifrado de contenido (CEK), el vector de inicialización JWE y el valor de datos de autenticación adicional (AAD), con el algoritmo de cifrado definido por el elemento de encabezado enc. El algoritmo definido por el elemento de encabezado enc debe ser un algoritmo simétrico de cifrado autenticado con datos asociados (AEAD).
- El elemento final del token es el valor codificado en base64url de la etiqueta autenticada JWE. Este valor se produce durante el proceso de cifrado AEAD, junto con el texto cifrado y asegura la integridad del texto cifrado y los datos autenticados adicionales (AAD)

2.4 Proceso de firma en la serialización compacta del JWE

A continuación, se describe el proceso de cifrado de un JWE bajo la serialización compacta:

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

- Se identifica el modo de administración de claves mediante el algoritmo utilizado para determinar el valor de Clave de cifrado de contenido (CEK). Este algoritmo está definido por el elemento alg en el encabezado JOSE.
- Se calcula el CEK y la clave cifrada JWE en función del modo de administración de claves. El CEK se utiliza más tarde para cifrar la carga útil JSON.
- Se calcula el valor codificado en base64url de la clave cifrada JWE, que se generó en el paso anterior. Este es el segundo elemento del token JWE.
- Se genera un valor aleatorio para el vector de inicialización JWE. Independientemente de la técnica de serialización, el token JWE llevará el valor del valor codificado en base64url del vector de inicialización JWE (este es el tercer elemento del token JWT).
- Si se necesita compresión de token, la carga útil JSON en texto sin formato debe comprimirse siguiendo el algoritmo de compresión definido en el elemento de encabezado zip (no es el caso).
- Se construye la representación JSON del encabezado JOSE y se obtiene el valor codificado en base64url del encabezado JOSE con codificación UTF8 (este es el primer elemento del token JWE).
- Para cifrar la carga útil JSON, se utiliza la CEK, el vector de inicialización JWE y los datos autenticados adicionales (AAD). El ADD se obtiene calculando el valor ASCII del encabezado JOSE codificado del paso anterior.
- Utilizando la CEK, el vector de inicialización JWE y los datos autenticados adicionales (AAD) se cifra la carga útil JSON comprimida (del paso anterior), siguiendo el algoritmo de cifrado de contenido definido por el elemento de encabezado enc header. El algoritmo definido por el elemento de encabezado enc es un algoritmo AEAD y, después del proceso de cifrado, produce el texto cifrado y la etiqueta de autenticación.
- A continuación, se calcula el valor codificado en base64url del texto cifrado. Este es el cuarto elemento del token JWE.
- Por último, se calcula el valor codificado en base64url de la etiqueta de autenticación. Este es el quinto elemento del token JWE.

2.5 Proceso de intercambio de los datos

El envío del token generado de los dos puntos anteriores se enviará en el cuerpo de la petición HTTP.

2.6 Ejemplo de mensajes

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

Mensaje serializado:

eyJ-raWQiOiJ0ZXN0LXNwZi1zdWJqZWN0LTAwMDEiLCJlbmMiOiJBMjU2R0NNIiwiYWxnIjoiUlNBLU9BRVAtMjU2In0.pVze_KduVP6Ih93GiliBoO8pi_jhKI1c1wbTKHj3v--KnBk1mkBgxwIdkq1e1WYHuXVkwa9fXTFFTO4JWyD0IcD4PwIbECzrSO3As-rRPgCwZ5MUpK01A8k85ryBMKgWv8GVA11VnvUYx-ni75OXNPAzIG8EhQN6eMNbldJ7Su-wBFvqZuu9BZDoDHT_JCzsWu3zXfCss6O_SxQYNlgBRZyDWjXzu_o-5LRUH0Uvwrmdhlvf7WLwXvlJ1Sw3M4hOSQ97cJMjCMBHFJ44EjwjTd_z4ZuGAKJi-L7LFL7sfE6IINaErVmrre91Wz5AyiDunPoy2JCpeOEE8XIWgOM0b_wQ.JWkgf2AqAfPybAK7.PHS0trBJ9EGiilJV_acRtDvBHIS2ooHgZxhs2P8UgIo1DD6-M6raObGyHkaIkBFbFcy7H3SUFA1y7E073BppWkZWPJXycdP6yDgvKFGmlOcXkXJajPH-NNuYI7DF6xtNqQJvd-ezr9_XGD0sjuOAq0WlpT7ji_KFbXxvDx61a-UYUkAx_QOi83q4XF-6oc4I20MtAfay86QrU1CCRZyxFSf_1yg1wKL3RspQRZFHiKxjzyTcXlPl.oExa_BvZnKm-dO7VO2_XfQ

Mensaje en claro:

```
{
 "enc": "A256GCM",
  "alg": "RSA-OAEP-256",
  "kid": {alias certificado},
  "error": [
    {
      "msg": {descripción del error},
      "cod": {código del error}
    }
  ],},
{
  "sub": {BIC entidad emisora},
  "aud": {BIC entidad receptora},
  "nbf": 1610719256,
  "data": {mensaje comprimido},
```

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

"iss": {*BIC entidad emisora*},

"exp": 1610719856,

"iat": 1610719256,

"jti": {*referencia*}

}

2.7 Códigos de error

A continuación, se describen los códigos de error generados del proceso de validación de firma y cifrado. Para ello, se ha definido un nuevo campo en la cabecera del token JWE denominado "err" de tipo array. Este campo contiene, a su vez, dos campos. El primero de ellos ("cod") indica el código de error y el segundo ("msg") contiene una breve descripción del error. En el siguiente listado se detallan los diferentes errores:

- IP no autorizada: HTTP 401
- Error de validación del token: HTTP 400 / cod 453
- Resto de errores: HTTP400 / cod 500

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

3. GESTIÓN DE CLAVES Y CERTIFICADOS

3.1 Características de las claves

Tanto para los servicios SOAP, como para los servicios REST, se usarán para el cifrado y firma un par de claves (pública/privada). Se utilizará el mismo par de claves tanto para la firma del documento, como para el cifrado la clave secreta que cifra los datos. Las características de las claves a generar son:

- Algoritmo RSA con longitud de clave de 2048 bits
- Versión 3
- Algoritmo de firma SHA-256 con RSA
- 2 año de validez para el par de claves

3.2 Intercambio de certificados

Para el intercambio de la parte pública de las claves generadas, que será necesaria para validar la firma y descifrar los mensajes, es necesario que ambos extremos en la comunicación, entidad e Iberpay, conozcan dicho elemento. Este intercambio se hará vía Servicio Web en el que cada extremo expondrá un servicio para permitir el intercambio de la clave pública. Dicho servicio será en formato SOAP o REST y tendrá las siguientes características:

- Nombre: WS_ENVIO_CERTIFICADO
- Campos:
  - Entidad: campo en formato String en el que se indicará el BIC de que entidad proviene el certificado.
  - Alias: campo en formato String que indica el nombre del certificado enviado. Debe seguir el patrón *[entorno]-[servicio]-[nombre entidad]-[secuencia(4 posiciones)]* donde:
    - entorno: los valores admitidos son *test* y *prod*.
    - servicio: puede contener los valores *spf* (Servicio de Prevención del Fraude), *tic* (Servicio de Titularidad de Cunetas) o trc (Servicio de Traslado de Cuentas).
    - nombre entidad: nombre de la entidad de la que es propiedad el certificado.
    - secuencia: contador de 4 dígitos.
  - Acción: campo en formato String que la acción a hacer con el certificado. Estas acciones puedes ser dos:
    - A (Actualizar): se actualiza el certificado que está en vigor con el que se indica en el campo "Document".

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

- R (revocar): revoca el certificado que se indique en el campo "Alias".

o Document: es un Array de Bytes donde estará incrustado el certificado (componente pública). Dicho certificado debe estar en formato X.509 y codificado en Base64.

3.3 Proceso de intercambio de certificados

Proceso de intercambio y renovación de los certificados:

El proceso de intercambio de los certificados entre las entidades e Iberpay se realizará de la siguiente forma:

- Iberpay generará un nuevo par de claves y enviará la parte publica a la entidad mediante el Servicio Web desplegado para este fin. En el campo "Acción" deberá rellenarse con el valor "A".
- La entidad almacenará el certificado según sus políticas de seguridad.
- La entidad generará un nuevo par de claves y enviará la parte publica a Iberpay mediante el Servicio Web desplegado para este fin. En el campo "Acción" deberá rellenarse con el valor "A".
- Iberpay almacenará el certificado según sus políticas de seguridad.
- En el momento que se complete el intercambio, los certificados deberán quedar activos.
- En caso de renovación, se realizará mediante el proceso de intercambio estándar. Los certificados quedaran activos en el momento del intercambio.

Proceso de revocación de los certificados:

El proceso de revocación de los certificados entre las entidades e Iberpay se realizará de la siguiente forma:

- El extremo que necesite revocar un certificado antes de la fecha de la expiración deberá enviar el certificado mediante el Servicio Web habilitado indicando en el campo "Alias" el certificado a revocar y en el campo "Accion" el valor "R".
- A continuación, el extremo que ha iniciado el proceso de revocación deberá enviar un nuevo certificado válido a través del Servicio Web habilitado.

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

4. MEDIDAS DE CONTINGENCIA

Iberpay contempla la desactivación manual de los mecanismos de firma y cifrado en caso de cualquier anomalía o error en cualquiera de los Servicios Web que hagan uso de ellas. Asimismo, las entidades tendrán que establecer las medidas oportunas en su extremo para intercambiar mensajes sin firma ni cifrado en caso de contingencia.

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay)

© Sociedad Española de Sistemas de Pago, S.A. (Iberpay).
Reservados todos los derechos.