PRETA Open Banking Europe: JSON Web Signature Profile for Open Banking

Open Banking Europe - providing collaborative services to support PSD2 XS2A, in partnership with the financial industry

Version: 000 003 Revised draft for review

Date: 20 February 2020

Contents

1. About Open Banking Europe

1.1. Purpose

The revised Payment Services Directive (PSD2) came into force in January 2018, with a requirement deadline of 14 September 2019 to implement Strong Customer Authentication (SCA). At this point, all regulated entities (Payment Service Providers) had to ensure that they individually complied with PSD2 and the Regulatory Technical Standards (RTS) set out by the European Banking Authority (EBA).

There is a clear regulatory expectation that the financial industry will organise itself to make sure that the implemented solutions for PSD2 are interoperable. However, at the time of writing, there remains a number of outstanding activities required to successfully achieve this expectation.

Open Banking Europe has been launched to support Account Servicing Payment Service Providers (ASPSPs) and Third Party Providers (TPPs) in meeting the Access to Account (XS2A) requirements of PSD2 and to facilitate the wider aims of Open Banking.

1.2. History

PRETA S.A.S. was created in 2013 to develop and innovate market competitive services in digital payment and identity solutions. The company is a wholly owned subsidiary of EBA CLEARING, a provider of pan-European payment solutions currently owned by shareholder banks.

Following a series of stakeholder consultations that started in 2016 to determine industry requirements, PRETA launched Open Banking Europe to build a PSD2 Directory solution to support PSPs and TPPs in meeting the PSD2 XS2A requirements. The Open Banking Europe Directory Service was released in January 2019, providing a single, standardised reference point for banks to accurately identify which TPPs are authorised to access their interfaces and which roles and services they are authorised to perform on behalf of their customers. Additionally, a Transparency Directory has been developed to help TPPs understand developer portals, and to help Account Servicing Payment Services Providers (ASPSPs) understand TPP brands. Open Banking Europe continues to work with stakeholders on a range of initiatives to facilitate a greater understanding of Open Banking and enable collaboration between interested parties.

1.3. Audience

Open Banking Europe is aimed at the following audiences:

- Competent Authorities
- Payment Service Providers (PSPs), including:
  - Account Servicing Payment Services Providers (ASPSPs)
  - Third Party Providers (TPPs)
- Qualified Trust Service Providers (QTSPs)
- Service Providers, Solution Providers, and relevant consultancies

1.4. Disclaimer

Whilst care has been taken to ensure that the information contained in this document is true and correct at the time of publication, there are still clarifications needed around PSD2's scope and implementation and this may impact on the accuracy of the information contained within this document.

As such, Open Banking Europe cannot guarantee the accuracy or reliability of any information contained within this document at the time of reading, or that it is suitable for your intended use.

2. General Introduction

2.1. Background

OBE brought together a group of experts from the PSD2 API communities with experts on signature formats from ETSI. The group carried out a survey of the current approaches to secure communications for PSD2 based on EU Qualified Certificates as required under the EU "regulatory technical standards for strong customer authentication and common and secure open standards of communication". As a result of the survey it was found that there were two basic approaches taken. About half API communities used JSON Web Signatures to protect the payload, whilst the other half use HTTP Signatures based on a draft specification originally authored by Cavage [27]. HTTP signatures were chosen primarily because of its ability to protect HTTP header information. As a result, it was agreed to produce a common specification of how to protect PSD2 payloads which brings together the JSON Web Signatures with the ability of HTTP Signatures to protect HTTP header information. It was also decided to align the specification with the ETSI "JAdES" specification [24] currently under development for advanced electronic signatures and seals in line with the EU eIDAS regulation. The present document is this common specification.

2.2. Scope

The present document defines a profile of JSON Web Signature (JWS hereinafter), as defined in RFC 7515[2] in support of secure communications under the Payment Services Directive 2015/2366 [16] (PSD2). In particular, it is aimed at supporting the secure communications between payment service providers using qualified certificates for electronic seals, (Article 3(30) of Regulation (EU) No 910/2014), as required under Commission Delegated Regulation (EU) 2018/389 [15] Article 34.

ETSI is developing a standard for JWS which includes the special features already in other ETSI standards for AdES digital signatures (see references [17] to [22]) and is aligned with Regulation (EU) No 910/2014, to be called JAdES [24] (to be ETSI TS 119 182-1). The current profile is aligned with the basic (B-B) level of JAdES and makes use of JWS extensions defined in JAdES.

2.3. Terminology

The following terms are used in the present document:

- *HTTP body*: the payload body carried by HTTP protocol RFC 7231 [23].
- *HTTP Header*: the request and response header fields carried by HTTP protocol RFC 7231 [23].
- *JSON Web Signature:* the whole JSON object including the *JWS Protected Header* and *JWS Signature Value.*

  Note: In this profile the payload is detached from the JSON Web Signature.
- *JWS Protected Header*: the collection of JSON Object Signing and Encryption (JOSE) header fields as defined in RFC 7515 [2] which are protected by the *JWS Signature Value*.
- *JWS Signature Value*: the digital signature cryptographic value calculated over a sequence of octets derived from the *JWS Protected Header* and *Data to be Signed*.
- *Data to be Signed*: the data used as input to the creation of the *JSON Web Signature*
- *Base64URL* an encoding of binary data into a character string using the URL- and filename-safe character set as defined in clause 2 of RFC 7515 [2].
- *UTC* the coordinated universal time standard.

  Note: This is equivalent to Greenwich Mean Time.
- *JAdES* JSON Web Signature following ETSI general AdES requirements as defined in ETSI TS 119 182-1 [24].

2.4. Design Principles

The profile defined in the present document is designed to meet the following requirements:

DESIGN-PRINCIPLE#1: The signatures compliant with this profile support the use of qualified certificates for electronic seals in line with Commission Delegated Regulation (EU) 2018/389 [15].

DESIGN-PRINCIPLE#2: The profile is aligned with JAdES [24] baseline digital signatures as specified by ETSI.

DESIGN-PRINCIPLE#3: The signature protects an *HTTP body* and optionally selected *HTTP Header* fields.

DESIGN-PRINCIPLE#4: A single signature is to be carried in an *HTTP Header* detached from the payload.

DESIGN-PRINCIPLE#5: The signature is as transparent as possible to any intermediate device that they may traverse when they are exchanged between parties (firewalls, front-ends, relays, etc).

DESIGN-PRINCIPLE#6: The profile aims to maximise interoperability.

DESIGN-PRINCIPLE#7: No restrictions are imposed to the contents of the signed payloads. It can be used to protect JSON, XML ISO 20022 or any other form of data.

DESIGN-PRINCIPLE#8: *JSON Web Signature* headers and *HTTP Header* Header fields which are critical to the security of the exchange, as well as *HTTP body*, are protected such that they cannot be modified.

DESIGN-PRINCIPLE#9: This profile follows generally accepted security best practices.

DESIGN-PRINCIPLE#10: The profile defined is extensible.

DESIGN-PRINCIPLE#11: Signatures can be later used as evidence in court (i.e. are "non-repudiable").

DESIGN-PRINCIPLE#12: Selected HTTP header parameters can be signed without there being an *HTTP body* (e.g. for GET or DELETE requests)

DESIGN-PRINCIPLE#13: Signatures can be applied to HTTP requests as well as responses.

2.5. Relationship between this profile with RFC 7151 (JWS) and ETSI TS 119 182-1 (JAdES)

The profile defined in the present document is based on both IETF RFC 7515 [2] (JWS) and ETSI TS 119 182-1 [24] (JAdES) as follows:

1) RFC 7515 defines a selection of encodings and structures, as well as a set of header parameters which may be used for JSON Web Signatures
2) TS 119 182-1 is based on RFC 7515. It allows any of the RFC 7515 encodings and structures to be used. It places some restrictions on the use of RFC 7515 header parameters. It also defines further header parameters which may be used along with the RFC 7515. It also defines 4 "baseline levels" for use of the header parameters.
3) This profile uses a specific encoding and structure as defined in RFC 7515.
4) This profile uses header parameters defined in RFC 7515 as well as header parameters defined in TS 119 182-1.
5) This profile is aligned with the requirements in TS 119 182-1 for the baseline level "B-B" but imposes additional restrictions.
6) This profile also makes use of the following IETF RFCs associated with RFC 7515: RFC 7518 [4] and RFC 7797 [3].

Signatures compliant with requirements of this profile are also compliant with the requirements of JAdES according to ETSI TS 119 182-1 and with JSON Web Signature according to RFC 7515

3. JWS Encoding

3.1. Introduction

A *JSON Web Signature* as specified in RFC 7515 [2] consists of:

1. A JOSE Header,
2. *Data to be Signed* (not present if detached),
3. The *JWS Signature Value*.

In the present profile:

- the JOSE header is a *JWS Protected Header* and
- the *JSON Web Signature* is detached from *Data to be Signed* (see section 4 below).

RFC 7515: "JSON Web Signature (JWS)" [2] specifies two ways of encoding this information (referred to as serialisation):

- JWS Compact Serialisation
- JWS JSON Serialisation

With JWS Compact Serialisation:

- all the header parameters are protected (forming the *JWS Protected Header*);
- all JWS elements listed above are encoded in *Base64URL* which converts data to a transparent character string which avoids any characters which could cause problems in firewall, relays etc or conflict with filename or URL character restrictions;
- Uses UTF8 character encoding for the header information.

With JWS JSON Serialisation:

- the data is encoded using JSON object notation;
- unsigned header fields can be added to the *JSON Web Signature* after signing.
- A general structure can be used to allow multiple signatures against the same object.;
- A simplified flattened structure can be used if only one signature is carried

3.2. Requirements

REQUIREMENT-1: The *JWS Structure* shall be encoded using JWS Compact Serialisation.

3.3. Rationale

- The representation of the *JSON Web Signature* obtained with this serialisation is most transparent to any intermediate device that the *JSON Web Signature* may traverse when they are exchanged between parties (firewalls, front-ends, relays, etc), as stated in (DESIGN-PRINCIPLE#5). This is because all the components of the *JSON Web Signature* are *Base64URL* encoded.
- The representation of the *JSON Web Signature* obtained with this serialisation maximises interoperability (DESIGN-PRINCIPLE#6).
- This serialisation protects all *JSON Web Signature* header *JWS Protected Header* parameters (DESIGN-PRINCIPLE#8).

4. Detached JWS

4.1. Introduction

RFC 7515 [2] Appendix F defines an encoding of *JSON Web Signature* whereby the signed content (*Data to be Signed*) can be detached from the *JSON Web Signature*. See also section 6.3 on identification of *Data to be Signed*.

RFC 7797: "JSON Web Signature (JWS) Unencoded Payload Option" [3] defines an extension to the *JWS Protected Header*, "b64", which if set to false indicates that the *Data to be Signed* does not need to be re-encoded in *Base64URL*.

The *HTTP Header* field "x-jws-signature" is already used in several PSD2 APIs to carry the detached *JSON Web Signature*.

4.2. Requirements

REQUIREMENT-2: The JWS header shall include b64 header parameter, as defined in RFC 7797 [3], set to false.

REQUIREMENT -3: The JWS content (*Data to be Signed*) shall be detached from the signatures as defined in RFC 7515 [2] Appendix F.

REQUIREMENT -4: The JWS structure shall be carried in *HTTP header* field named "x-jws-signature".

4.3. Rationale

- The signature is to be detached from the *HTTP body*. (DESIGN-PRINCIPLE#4).
- The encoding of the payload is unrestricted. (DESIGN-PRINCIPLE#7).
- The use of the x-jws-signature facilitates interoperability with existing implementations (DESIGN-PRINCIPLE#6)

5. RFC 7515 Header Parameters

5.1. General Overview

5.1.1. Introduction

JWS is one of a set of specifications for security of JSON. This also includes JSON Web tokens (JWT), JSON Web Encryption (JWE) and JSON Web Keys (JWK). All these specifications share a common header structure call JSON Object Signing and Encryption (JOSE). JWS uses this shared JOSE structure.

In this profile all the header parameters contained in *JWS Protected Header* are protected by the signature, and consequently the *JWS Protected Header* is equal to the JWS protected header.

RFC 7515: "JSON Web Signature (JWS)" [2] identifies three classes of *JWS Protected Header* parameters.

a) Registered Header Parameters: Header parameters defined in RFC 7515 for use in JWS. The header names are registered as specified in section 9.1 of RFC 7515.

b) Public Header Parameters: Header parameters defined in other public specifications. ETSI is defining additional header parameters in JAdES which it is planned to register as specified in section 9.1 of RFC 7515.

c) Private Header Parameters: Header parameters which agreed to use within a closed community. These names may be subject to collision and so should be used with caution. Until formally registered the ETSI defined parameters are to be treated as private.

This section specifies the usage of the RFC 7515 registered header parameters.

5.1.2. Requirements

RECOMMENDATION-05:API Communities may define their own private header parameters but as stated in RFC 7515 this may cause problems. API communities which have requirements for additional JWS *JOSE Header* parameters which are considered necessary are encouraged to inform ETSI and OBE so that this can be made publicly known. Steps should also be taken to register any additional header parameters as specified in section 9.1 of RFC 7515.

5.1.3. Rationale

- The profile is aimed to be extensible DESIGN-PRINCIPLE#10 whilst also maximising interoperability DESIGN-PRINCIPLE#6.

5.2. Parameters for Identifying Cryptographic Algorithm: Header Parameter "alg"

5.2.1. Introduction

The "alg" Header parameter identifies the cryptographic algorithm used to create the *JWS Signature Value*. See section 6 for further details on recommended algorithms.

5.2.2. Requirements

REQUIREMENT-6: The "alg" Header parameter shall be present. "none", as defined in section 3.6 of RFC 7518 , shall not be used.

5.2.3. Rationale

- This counters attacks on the selection of the cryptographic algorithm used for creating the *JWS Signature Value* (DESIGN-PRINCIPLE#8).

5.3. Parameters for Binding the Signing Certificate

5.3.1. Overview of Mechanisms

RFC 7515 [2] defines a number of Header parameters supporting mechanisms that can be used for exchanging information on the JWS signing key or the signing certificate. The public key used for validating a signature can be provided through one of the following parameters:

- jku: a URI pointing to the resource where the public key may be retrieved from
- jwk: the public key for validating the signature
- kid: an identifier of the public key for validating the signature
- x5u: a URI pointing to the resource where the X509 signing certificate (with or without the certification path) may be retrieved from
- x5c: X.509 public key certificate or certificate chain corresponding to the key used to create the JSON Web Signature.
- x5t: digest of the X509 signing certificate using SHA1
- x5t#S256: digest of the X509 signing certificate using SHA 256.

5.3.2. Requirements

REQUIREMENT-7: Either the "x5c" or "x5t#S256" header parameter shall be present. Both the "x5c" and "x5t256" header parameters shall not be present in the same *JWS JOSE Header*. The certificate identified by one or other of these header parameters shall be used for validation of the signature.

REQUIREMENT-8: The signing certificate shall be carried in the "x5c" header parameter unless there is a prior arrangement to use a certificate that has already been registered with parties relying of this signature.

REQUIREMENT-9: The "x5t#S256" hash shall be used if the header parameter "x5c" is not present.

REQUIREMENT-10: The "x5t" the header parameter shall not be used.

OPTION -11: The " x5c" header parameter may include the full certificate path up to the trust anchor. If present, this need not be used by the relying party to validate the signature.

OPTION -12: The "kid" header parameter may be used to further identify a certificate if "x5t#S256" is used. If present the kid shall contain IssuerSerial as defined in JAdES. If present, this need not be checked against the certificate used to validate the signature by the relying party.

OPTION -13: The "x5t#o" header parameter (as defined in JAdES see section 6.4) may be used to further identify a certificate if "x5t#S256" is used. If present, this need not be checked against the certificate used to validate the signature by the relying party.

Note: Guidance on use of cryptographic algorithms, including hashing algorithms, is given in section 7

OPTION -14: The "x5u" header parameter may be used to further identify a certificate if "x5t#S256" or "x5t#o" are used. If present, this need not be checked against the certificate used to validate the signature by the relying party.

5.3.3. Rationale

- This supports the use of qualified certificates for electronic seals in line with Commission Delegated Regulation (EU) 2018/389 [15] (DESIGN-PRINCIPLE#1).
- The use of SHA-1 as required for x5t is generally deprecated thus x5t#S256 using SHA-256 is recommended (DESIGN-PRINCIPLE#9).
- The current use of "kid" can be maintained by API communities but should not be relied upon for identifying the signing certificate (DESIGN-PRINCIPLE#6).

5.4. Parameter for Identifying the Type of JWS Structure

5.4.1. Introduction

The header parameter "typ" is used to identify that the general structure of header is the *JWS Protected Header* structure shared with similar JSON security specifications (see section 5.1) and also to identify whether compact serialisation or JSON serialisation is used. For JWS has two values, namely: "JOSE", or "JOSE+JSON" respectively.

5.4.2. Requirements

RECOMMENDATION-15: "typ" Header parameter should be set to "JOSE".

5.4.3. Rationale

- 'JOSE' aligns with the use of compact serialisation as required in section 3.

5.5. Parameter for Identifying the Type of the Payload: Parameter Header "cty"

5.5.1. Introduction

Clause 4.1.9 of RFC 7515 [2] states that the content of this Header parameter "is used by JWS applications to declare the media type of the secured content (the payload)".

This Header Parameter is useful in contexts where JWS applications may receive JSW Payloads of different types, each requiring a specific treatment.

It should be considered that, while "all media type values, subtype values and parameter names are case insensitive" for RFC 2045: "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies" [13], Header "parameter values are case sensitive unless otherwise specified for the specific parameter".

The *HTTP Header* Parameter Content-Type already defines the media type of the *HTTP body*.

5.5.2. Requirements

REQUIREMENT-16: The "cty" header parameter shall not be present.

5.5.3. Rationale

This requirement is better met through use of the *HTTP Header* Parameter Content-Type.

5.6. Parameter Identifying Other Critical Header Parameters

5.6.1. Introduction

Header parameters can be either critical or non-critical.

Applications unable to understand and/or process critical header parameters must reject the signature. Applications unable to understand and/or process non-critical header parameters can ignore them.

All the header parameters specified in RFC 7515 [2] are critical by definition.

To make any other header parameter defined outside RFC 7515 that need to be understood and processed, they must be listed within the "crit" header parameter. Thus, the header parameters required to be present in other sections need to be marked as critical.

5.6.2. Requirements

REQUIREMENT-17: The "crit" header parameter shall include the following header parameter names:

- b64 (see section 4.2)
- sigT (see section 6.2)
- sigD (see section 6.3)

Other non RFC 7515 headers should not be marked as critical.

5.6.3. Rationale:

This REQUIREMENT maximises interoperability (DESIGN-PRINCIPLE#6) but allows a degree of extensibility with the ability to include non-critical headers (DESIGN-PRINCIPLE#10).

5.7. Use of Other RFC 7515 Header Parameters

5.7.1. Introduction

The following JWS Header parameter defined in RFC 7515 are not addressed by requirements above:

- "jwk"
- "jku"

The use of this attribute is not considered appropriate to the use of PSD2 which requires the use of qualified certificates.

5.7.2. Requirements

REQUIREMENT-18: The following header parameter shall not be present: "jwk", "jku".

5.7.3. Rationale:

Use of other header parameters will hinder interoperability (DESIGN-PRINCIPLE#6).

6. ETSI JAdES Header Parameters

6.1. General Overview

ETSI is specifying a profile of RFC 7515 JWS, to support EU regulatory requirements referred to as JAdES in ETSI draft TS 119182 [24]. This profile includes the definition of additional *JSON Web Signature* public header parameters.

In this profile all the header parameters are contained in a *JWS Protected Header*.

This section specifies the usage of the JAdES public header parameters required by this profile.

6.2. Parameter for Signing Time

6.2.1. Introduction

In order to validate a signature subsequent to the transaction it is necessary to establish a "proof of existence" time at which the signing operation occurs (see ETSI TS 119 102-1 [25]). The claimed signing time, which is required to be included with the signature in JAdES, can be used as the basis for such a proof of existence.

The JAdES "sigT" header parameter contains the claimed signing time encoded using RFC3339 Internet time format for UTC without seconds (e.g. "2019-11-19T17:28:15Z").

The recipient of the signature HTTP request or response can further validate this against its local time on receipt of the message.

6.2.2. Requirements

REQUIREMENT-19: The JAdES "sigT" header parameter shall be present and set to the time that the signature was created. The time shall be indicated in UTC (ending in "Z") and shall indicate date and time to the second.

RECOMMENDATION-20:The recipient of an HTTP request / response should check the claimed signing time indicated by sigT against locally managed time and reject the signature if it is outside the maximum time window expected for transactions. This time window for transactions should be less than 4 hours.

6.2.3. Rationale

- Signing time is considered to be a critical security parameter and so needs to be protected by the *JWS Signature Value* (DESIGN-PRINCIPLE#8).
- By cross checking against the HTTP transaction time and the recipients local time the claimed signing time can be confirmed, and the risks of replay attacks by intermediaries reduced.
- As the signing time is accepted by both parties it can be used as "proof of existence" of the signature supporting its "non-repudiability" DESIGN-PRINCIPLE#11.

6.3. Parameter for Identifying Data to be Signed

6.3.1. Introduction

The JAdES "sigD" header parameter contains:

mID: A URI which identifies the mechanism used to identify the *Data to be Signed*.

pars: Parameters of this mechanism.

The identification mechanism is defined in JAdES ("http: //uri.etsi.org/19182/HttpHeaders") which specifies how *HTTP Header* fields are used to create the *Data to be Signed*. This mechanism follows the string

construction procedures to create the *Data to be Signed* as defined in draft-cavage-http-signatures-10 [27] section 2.3. The parameter values in sigD identify the *HTTP Header*s which are used to create the *Data to be Signed*. This identification mechanism recognises "(request-target)" as a special case

The *Data to be Signed* is detached from the *JSON Web Signature* as described in section 4.

In current implementations, existing prior to the publication of the present document, the *HTTP Header* field "x-jws-signature" is used to carry a detached *JSON Web Signature* with *Data to be Signed* being the *HTTP body*.

The *HTTP Header* "Digest" as defined in RFC 3230[26] applied to the message body provides a digest of the *HTTP body* which can be used to sign the *HTTP body*.

6.3.2. Requirements

REQUIREMENT-21: The JAdES "sigD" header parameter shall be present with "mID" set to http://uri.etsi.org/19182/HttpHeaders.

REQUIREMENT-22: The JAdES "sigD" "Pars" shall include the following *HTTP Header* field name:

- "Digest" as defined in RFC 3230[26] applied to the *HTTP body*. If the HTTP Body is not present the "Digest" header shall contain the hash of an empty bytelist.

Note: In case of a multipart message the same method is used to calculate the digest. I.e. a hash of the (whole) *HTTP body*is calculated including all parts of the multipart message as well as the separators."

RECOMENDATION-23: The JAdES "sigD" "Pars" should include the following *HTTP Header* field name:

- "(request-target)" for HTTP Requests
- "Content-Type" if present
- "Content Encoding" if present

RECOMENDATION-24: Parties relying on JWS signatures, where the sigD is not present, should consider the *HTTP Body* as the *Data to be Signed*

6.3.3. Rationale

- Use of sigD with transform "http://uri.etsi.org/19182/HttpHeaders" supports protection of *HTTP Header*s (DESIGN-PRINCIPLE#3)
- *HTTP Header* "Digest" based on RFC 3230[26] enables protection of the *HTTP body* (DESIGN-PRINCIPLE#3)
- *HTTP Header* fields which have a security impact should be included in the *Data to be Signed* DESIGN-PRINCIPLE#8)
- Recognition of JWS without the sigD facilitates backward compatibility with existing JWS implementations which do not protect any HTTP header information (DESIGN-PRINCIPLE#6).

6.4. Algorithm Agile "x5t#o" certificate hash

6.4.1. Introduction

In the past it has been found that hash algorithms have been susceptible to attack. It cannot be guaranteed that the same does not apply to the SHA 256 hashing algorithms.

The JAdES specification defines an alternative to the "x5t#S256" which includes the identifier of alternative hashing algorithm to enable the algorithm to be changed without changing the JWS protocol.

Note: x5t#S256 is still required to be present as defined in section 5.3.

6.4.2. Requirements

RECOMMENDATION 025: If an alternative to SHA-256 needs to be used the JAdES defined x5t#o should be supported.

6.4.3. Rationale

The use of x5t#o facilitates use of alternative hash algorithms to protect the certificate (DESIGN-PRINCIPLE#10).

6.5. Use of other JAdES header parameters

6.5.1. Introduction

TS 119 182-1 (JAdES) defines a number of protected header parameters which can be used to further enhance the signature but it is not expected that all implementations of this profile will support these header parameters.

As this profile uses compact serialisation (see section 3) it is not possible to use JAdES unprotected header parameters.

6.5.2. Requirements

REQUIREMENT 026: Any JAdES protected header parameters shall not be marked as critical except as indicated in section 5.6.

6.5.3. Rationale

This enables the profile to be extensible (DESIGN-PRINCIPLE#10) whilst maintaining interoperability (DESIGN-PRINCIPLE#6).

7. Algorithms & Key Lengths

7.1. Introduction

As described in section 5.1.2, the *JWS Protected Header* includes the “alg” header parameter which indicates the cryptographic algorithm used for a signature. Also, the "x5t#S256" and the "x5t#o" header parameters make use of hashing algorithms. This section provides guidance on the algorithms to be used.

RFC 7515: "JSON Web Signature (JWS)" [2] is complemented by RFC 7518: "JSON Web Algorithms (JWA)" [4], which registers a wide variety of "cryptographic algorithms and identifiers to be used with the JSON Web Signatures" and "defines several IANA registries for these identifiers".

RFC 7518: "JSON Web Algorithms (JWA)" [4] defines identifiers for the most relevant cryptographic algorithms for being used within JSON Web Signatures.

ETSI has published and regularly updates ETSI TS 119 312: "Electronic Signatures and Infrastructures (ESI); Cryptographic Suites" [14] (which when the present document was written was in its version v1.3.1 (February of 2019). This identifies the recommended algorithms and key length for electronic seals.

7.2. Requirements

RECOMMENDATION-027: It is recommended to use algorithms that are listed in both RFC 7518 [4] and ETSI TS 119 312 [14].

RECOMMENDATION-028: It is recommended to regularly review the policy on use of cryptographic algorithms, including the recommendations provided by ETSI TS 119 312 [14] and RFC 7518 [4], to take account of any potential weaknesses identified in the cryptographic algorithms used and the need of replacing them by stronger ones.

7.3. Rationale

- These recommendations will ensure the usage of secure cryptographic algorithms (as it has been already mentioned ETSI TS 119 312[14] is regularly updated) (DESIGN-PRINCIPLE#9).

8. References

[1] RFC 8259: "The JavaScript Object Notation (JSON) Data Interchange Format". December 2017.

[2] RFC 7515: "JSON Web Signature (JWS)". May 2015.

[3] RFC 7797: "JSON Web Signature (JWS) Unencoded Payload Option". February 2016.

[4] RFC 7518: "JSON Web Algorithms (JWA)". May 2015.

[5] RFC 7519: "JSON Web Token (JWT)". May 2015.

[6] RFC 7516: "JSON Web Encryption (JWE)". May 2015.

[7] Regulation (EU) No 910/2014 of the European Parliament and of the Council on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC. OJ L 257, 28.08.2014, p. 73-114. Available at:
https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32014R0910&from=EN

[8] RFC 4648: "The Base16, Base32, and Base64 Data Encodings". October 2006.

[9] RFC 3629: "UTF-8, a transformation format of ISO 10646". November 2003.

[10] UNICODE: "The Unicode Standard".

[11] RFC 5280: "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile". May 2008.

[12] COMMISSION IMPLEMENTING DECISION (EU) 2015/1505 of 8 September 2015 laying down technical specifications and formats relating to trusted lists pursuant to Article 22(5) of Regulation (EU) No 910/2014 of the European Parliament and of the Council on electronic identification and trust services for electronic transactions in the internal market. Available at:
https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32015D1505&from=EN

[13] RFC 2045: "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies". November 1996.

[14] ETSI TS 119 312: "Electronic Signatures and Infrastructures (ESI); Cryptographic Suites" v1.3.1. February 2009.

[15] COMMISSION DELEGATED REGULATION (EU) 2018/389 of 27 November 2017 supplementing Directive (EU) 2015/2366 of the European Parliament and of the Council with regard to regulatory technical standards for strong customer authentication and common and secure open standards of communication. Available at:
https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32018R0389&rid=7

[16] DIRECTIVE (EU) 2015/2366 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 25 November 2015 on payment services in the internal market, amending Directives 2002/65/EC, 2009/110/EC and 2013/36/EU and Regulation (EU) No 1093/2010, and repealing Directive 2007/64/EC. Available at:
https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32015L2366&from=EN

[17] ETSI EN 319 122-1: "Electronic Signatures and Infrastructures (ESI); CAdES digital signatures; Part 1: Building blocks and CAdES baseline signatures".

[18] ETSI EN 319 132-1: "Electronic Signatures and Infrastructures (ESI); XAdES digital signatures; Part 1: Building blocks and CAdES baseline signatures".

[19] ETSI EN 319 142-1: "Electronic Signatures and Infrastructures (ESI); PAdES digital signatures; Part 1: Building blocks and PAdES baseline signatures".

[20] ETSI EN 319 122-2: "Electronic Signatures and Infrastructures (ESI); CAdES digital signatures; Part 2: Extended CAdES signatures".

[21] ETSI EN 319 132-2: "Electronic Signatures and Infrastructures (ESI); XAdES digital signatures; Part 2: Extended XAdES signatures".

[22] ETSI EN 319 142-2: "Electronic Signatures and Infrastructures (ESI); PAdES digital signatures; Part 2: Additional PAdES signatures profiles".

[23] IETF RFC 7231: "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content"

[24] ETSI TS 119 182-1 "Electronic Signatures and Infrastructures (ESI); JAdES digital signatures; Part 1: Building blocks and JAdES baseline signatures (under development)"

[25] ETSI TS 119 102-1" Electronic Signatures and Infrastructures (ESI); Procedures for Creation and Validation of AdES Digital Signatures; Part 1: Creation and Validation"

[26] IETF RFC 3230 "Instance Digests in HTTP" January 2002"

[27] Internet Draft draft-cavage-http-signatures-10 " Signing HTTP Messages"

Annex A: Examples

[Examples to be added.]

Annex B: Summary of conformance requirements - Informative

See referenced section for specific requirements.

Field	Signer includes in	Relying Party to process if present	Section ref
alg	M	M	5.2
x5c	C	M	5.3
x5c#256	C	M	5.3
kid	O	O	5.3
x5t#o	C	O	5.3, 6.4
x5u	O	O	5.3
typ	O	O	5.4
cty	- M	-	5.5
crt	M	M	5.6
jwk	- M	-	5.7
jku	- M	-	5.7
sigT	M	R	6.2
sigD	M	M	6.3

M = Mandatory

R = Recommended

C = Conditional

O = Optional

- M = Mandatory not present