Internet Engineering Task Force (IETF)
Request for Comments: 8225
Category: Standards Track
ISSN: 2070-1721

C. Wendt
Comcast
J. Peterson
Neustar Inc.

February 2018

PASSporT: Personal Assertion Token

Abstract

This document defines a method for creating and validating a token that cryptographically verifies an originating identity or, more generally, a URI or telephone number representing the originator of personal communications. The Personal Assertion Token, PASSporT, is cryptographically signed to protect the integrity of the identity of the originator and to verify the assertion of the identity information at the destination. The cryptographic signature is defined with the intention that it can confidently verify the originating persona even when the signature is sent to the destination party over an insecure channel. PASSporT is particularly useful for many personal-communications applications over IP networks and other multi-hop interconnection scenarios where the originating and destination parties may not have a direct trusted relationship.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8225.

Table of Contents

1. Introduction 4
2. Terminology 4
3. PASSporT Overview 5
4. PASSporT Header 6
4.1. "typ" (Type) Header Parameter 6
4.2. "alg" (Algorithm) Header Parameter 6
4.3. "x5u" (X.509 URL) Header Parameter 6
4.4. Example PASSporT Header 7
5. PASSporT Payload 7
5.1. JWT-Defined Claims 7
5.1.1. "iat" (Issued At) Claim 7
5.2. PASSporT-Specific Claims 8
5.2.1. Originating and Destination Identity Claims 8
5.2.2. "mky" (Media Key) Claim 10
6. PASSporT Signature 11
7. Compact Form of PASSporT 12
7.1. Example Compact Form of PASSporT 13
8. Extending PASSporT 13
8.1. "ppt" (PASSporT) Header Parameter 13
8.2. Example Extended PASSporT Header 14
8.3. Extended PASSporT Claims 14
9. Deterministic JSON Serialization 15
9.1. Example PASSporT Deterministic JSON Form 16
10. Security Considerations 17
10.1. Avoidance of Replay and Cut-and-Paste Attacks 17
10.2. Solution Considerations 18
11. IANA Considerations 18
11.1. Media Type Registration 18
11.2. Registrations in "JSON Web Token Claims" 19
11.3. Registration in "JSON Web Signature and Encryption Header Parameters" 20
11.4. PASSporT Extensions Registry 20
12. References 20
12.1. Normative References 20
12.2. Informative References 22
Appendix A. Example ES256-Based PASSporT JWS Serialization and Signature 23
A.1. X.509 Private Key in PKCS #8 Format for ES256 Example 24
A.2. X.509 Public Key for ES256 Example 25
Acknowledgments 25
Authors' Addresses 25

1. Introduction

In today's IP-enabled telecommunications world, there is a growing concern about the ability to trust incoming invitations for communications sessions, including video, voice, and messaging [RFC7340]. As an example, modern telephone networks provide the ability to spoof the calling party's telephone number for many legitimate purposes, including providing network features and services on behalf of a legitimate telephone number. However, as we have seen, bad actors have taken advantage of this ability for illegitimate and fraudulent purposes meant to trick telephone users into believing that they are someone they are not. This problem can be extended to many emerging forms of personal communications.

This document defines a method for creating and validating a token that cryptographically verifies an originating identity or, more generally, a URI or telephone number representing the originator of personal communications. Through the extensions defined in Section 8 of this document, other information relevant to the personal communications can also be added to the token. The goal of PASSporT is to provide a common framework for signing information related to the originating identity in an extensible way. Additionally, this functionality is independent of any specific call logic for personal-communications signaling, so that the assertion of information related to the originating identity can be implemented in a flexible way and can be used in such applications as end-to-end applications that require different signaling protocols or gateways between different communications systems. It is anticipated that guidance specific to the signaling protocol will be provided in other related documents and specifications to specify how to use and transport PASSporTs; however, this is intentionally out of scope for this document.

[RFC8224] provides details of the use of PASSporT within the SIP [RFC3261] signaling protocol for the signing and verification of telephone numbers and SIP URIs.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. PASSporT Overview

"JSON Web Token (JWT)" [RFC7519], "JSON Web Signature (JWS)" [RFC7515], and other related specifications define a standard token format that can be used as a way of encapsulating claimed or asserted information with an associated digital signature using X.509-based certificates. JWT provides a set of claims in JSON format that can conveniently accommodate asserted originating-identity information and that are easily extensible for use in the extension mechanisms defined below. Additionally, JWS provides a path for updating methods and cryptographic algorithms used for the associated digital signatures.

JWS defines the use of JSON data structures in a specified canonical format for signing data corresponding to the JSON Object Signing and Encryption (JOSE) Header, JWS Payload, and JWS Signature. JWT defines a set of claims that are represented by specified JSON objects that can be extended with custom keys for specific applications. The next sections define the header and claims that MUST be minimally used with JWT and JWS for PASSporT.

PASSporT specifically uses this token format and defines claims that convey the identity of the origination and destination of personal communications. The primary value asserted in a PASSporT object is the originating identity representing the identity of the calling party or the initiator of a personal-communications session. The signer of a PASSporT object may or may not correspond to the originating identity. For a given application's use or using protocol of PASSporT, the creation of the PASSporT object is performed by an entity that is authoritative to assert the caller's identity. This authority is represented by the certificate credentials and the signature, and the PASSporT object is created and initiated to the destination(s) per the application's choice of authoritative point(s) in the network. For example, the PASSporT object could be created at a device that has authenticated with a user or at a network entity with an authenticated trust relationship with that device and its user. Destination identities represent the intended destination of the personal communications, i.e., the identity(s) being called by the caller. The destination point or points determined by the application need to have the capability to verify the PASSporT and the digital signature. The PASSporT-associated certificate is used to validate the authority of the originating signer, generally via a certificate chain to the trust anchor for that application.

4. PASSporT Header

The JWS token header is a JOSE Header ([RFC7515], Section 4) that defines the type and encryption algorithm used in the token.

The PASSporT header should include, at a minimum, the Header Parameters defined in the next three subsections.

4.1. "typ" (Type) Header Parameter

The "typ" (Type) Header Parameter is defined in JWS ([RFC7515], Section 4.1.9) to declare the media type of the complete JWS.

For the PASSporT, the "typ" header MUST be the string "passport". This signifies that the encoded token is a JWT of type "passport".

4.2. "alg" (Algorithm) Header Parameter

The "alg" (Algorithm) Header Parameter is defined in JWS ([RFC7515], Section 4.1.1). This definition includes the ability to specify the use of a cryptographic algorithm for the signature part of the JWS. It also refers to a list of defined "alg" values as part of a registry established by JSON Web Algorithms (JWA) ([RFC7518], Section 3.1).

For the creation and verification of PASSporTs and their digital signatures, implementations MUST support ES256 as defined in JWA ([RFC7518], Section 3.4). Implementations MAY support other algorithms registered in the "JSON Web Signature and Encryption Algorithms" registry created by [RFC7518]. The contents of that registry may be updated in the future, depending on cryptographic strength requirements guided by current security best practices. The mandatory-to-support algorithm for PASSporTs may likewise be updated in future updates to this document.

Implementations of PASSporT digital signatures using ES256 as defined above SHOULD use the deterministic Elliptic Curve Digital Signature Algorithm (ECDSA) if or when supported for the reasons stated in [RFC6979].

4.3. "x5u" (X.509 URL) Header Parameter

As defined in JWS ([RFC7515], Section 4.1.5), the "x5u" Header Parameter defines a URI [RFC3986] referring to the resource for the X.509 public key certificate or certificate chain [RFC5280] corresponding to the key used to digitally sign the JWS. Generally, as defined in JWS ([RFC7515], Section 4.1.5), this would correspond to an HTTPS or DNSSEC resource using integrity protection.

4.4. Example PASSporT Header

An example of the header would be the following, including the specified passport type, ES256 algorithm, and a URI referencing the network location of the certificate needed to validate the PASSporT signature.

```
{
  "typ":"passport",
  "alg":"ES256",
  "x5u":"https://cert.example.org/passport.cer"
}
```

5. PASSporT Payload

The token claims consist of the information that needs to be verified at the destination party. These claims follow the definition of a JWT claim ([RFC7519], Section 4) and are encoded as defined by the JWS Payload ([RFC7515], Section 3).

PASSporT defines the use of a standard JWT-defined claim as well as custom claims corresponding to the two parties associated with personal communications -- the originator and destination, as detailed below.

For PASSporT, any claim names MUST use the ASCII character set. Any claim values can contain characters that are outside the ASCII range, consistent with the rules of creating a JWT Claims Set as defined in [RFC7519], Section 7.1.

5.1. JWT-Defined Claims

5.1.1. "iat" (Issued At) Claim

The JSON claim MUST include the "iat" (Issued At) claim ([RFC7519], Section 4.1.6). As defined, the "iat" claim should be set to the date and time of issuance of the JWT and MUST indicate the date and time of the origination of the personal communications. The time value should be of the NumericDate format as defined in [RFC7519], Section 2. This is included for securing the token against replay and cut-and-paste attacks, as explained further in Section 10 ("Security Considerations").

5.2. PASSporT-Specific Claims

5.2.1. Originating and Destination Identity Claims

The originating identity and the destination identity are represented by two claims that are required for PASSporT -- the "orig" and "dest" claims. Both "orig" and "dest" MUST contain claim values that are identity claim JSON objects where the child claim name represents an identity type and the claim value is the identity string, both defined in subsequent subsections. Currently, these identities can be represented as either telephone numbers or Uniform Resource Indicators (URIs).

The "orig" claim is a JSON object with the claim name of "orig" and a claim value that is a JSON object representing the asserted identity of any type (currently either "tn" or "uri") of the originator of the personal-communications signaling. There MUST be exactly one "orig" claim with exactly one identity claim object in a PASSporT object.

Note: As explained in Section 3, the originating identity represents the calling party and may or may not correspond to the authoritative signer of the token.

The "dest" claim is a JSON object with the claim name of "dest" and MUST have at least one identity claim object. The "dest" claim value is an array containing one or more identity claim JSON objects representing the destination identities of any type (currently "tn" or "uri"). If the "dest" claim value array contains both "tn" and "uri" claim names, the JSON object should list the "tn" array first and the "uri" array second. Within the "tn" and "uri" arrays, the identity strings should be put in lexicographical order, including the scheme-specific portion of the URI characters.

Note: As explained in Section 3, the destination identity represents the called party and may or may not correspond to the authoritative party verifying the token signature.

5.2.1.1. "tn" (Telephone Number) Identity

If the originating or destination identity is a telephone number, the claim name representing the identity MUST be "tn".

The claim value for the "tn" claim is the telephone number and MUST be canonicalized according to the procedures specified in [RFC8224], Section 8.3.

5.2.1.2. "uri" (URI) Identity

If any of the originating or destination identities is in the form of a URI as defined in [RFC3986], the claim name representing the identity MUST be "uri", and the claim value is the URI form of the identity.

5.2.1.3. Future Identity Forms

We recognize that in the future there may be other standard mechanisms for representing identities. The "orig" and "dest" claims currently support "tn" and "uri" but could be extended in the future to allow for other identity types with new IANA-registered unique types to represent these forms.

5.2.1.4. Examples

The following is an example of a single originator with telephone number identity +12155551212, to a single destination with URI identity "sip:alice@example.com":

```
{
  "dest":{"uri":["sip:alice@example.com"]},
  "iat":1443208345,
  "orig":{"tn":"12155551212"}
}
```

The following is an example of a single originator with telephone number identity +12155551212, to multiple destination identities with telephone number identity +12125551212 and two URI identities -- "sip:alice@example.com" and "sip:bob@example.com":

```
{
  "dest":{
    "tn":["12125551212"],
    "uri":["sip:alice@example.com",
      "sip:bob@example.net"]
  },
  "iat":1443208345,
  "orig":{"tn":"12155551212"}
}
```

5.2.2. "mky" (Media Key) Claim

Some protocols that use PASSporT may also want to protect media security keys delivered within their signaling in order to bind those keys to the identities established in the signaling layers. The "mky" claim is an optional PASSporT claim defining the assertion of media key fingerprints carried in the Session Description Protocol (SDP) [RFC4566] via the "a=fingerprint" attribute ([RFC4572], Section 5). This claim can support either a single fingerprint or multiple fingerprints appearing in a single SDP body corresponding to one or more media streams offered as defined in [RFC8122].

The "mky" claim MUST be formatted as a JSON object with an array that includes the "alg" and "dig" claims with the corresponding algorithm and hexadecimal values. If there is more than one fingerprint value associated with different media streams in SDP, the fingerprint values MUST be constructed as a JSON array denoted by square brackets ("[" and "]"). For the "dig" claim, the claim value MUST be the hash of the hexadecimal value without any colons.

The "mky" claim is a JSON object with a claim name of "mky" and a claim value of a JSON array denoted by brackets. The "mky" claim value JSON array MUST be constructed as follows:

1. Take each "a=fingerprint" line carried in the SDP.

2. Sort the lines based on the UTF-8 [RFC3629] encoding of the concatenation of the "alg" and "dig" claim value strings.

3. Encode the array in the order of the sorted lines, where each "mky" array element is a JSON object with two elements corresponding to the "alg" and "dig" objects, with "alg" first and "dig" second.

An example claim with the "mky" claim is as follows:

For an SDP offer that includes the following fingerprint values,

```
a=fingerprint:sha-256 4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:
5D:49:6B:19:E5:7C:AB:3E:4B:65:2E:7D:46:3F:54:42:CD:54:F1
a=fingerprint:sha-256 02:1A:CC:54:27:AB:EB:9C:53:3F:3E:4B:65
:2E:7D:46:3F:54:42:CD:54:F1:7A:03:A2:7D:F9:B0:7F:46:19:B2
```

the PASSporT Payload object would be:

```
{
  "dest":{"uri":["sip:alice@example.com"]},
  "iat":1443208345,
  "mky":[
    {
      "alg":"sha-256",
      "dig":"021ACC5427ABEB9C533F3E4B652E7D463F5442CD54
        F17A03A27DF9B07F4619B2"
    },
    {
      "alg":"sha-256",
      "dig":"4AADB9B13F82183B540212DF3E5D496B19E57C
        AB3E4B652E7D463F5442CD54F1"
    }
  ],
  "orig":{"tn":"12155551212"}
}
```

6. PASSporT Signature

The signature of the PASSporT is created as specified by JWS ([RFC7515], Section 5.1, Steps 1 through 6). PASSporT MUST use the JWS Protected Header. For the JWS Payload and the JWS Protected Header, however, the lexicographic ordering and whitespace rules described in Sections 4 and 5 of this document, and the JSON serialization rules in Section 9 of this document, MUST be followed.

Appendix A of this document has a detailed example of how to follow the steps to create the JWS Signature.

Step 7 of the JSON serialization procedure in [RFC7515], Section 5.1 is not supported for PASSporT.

[RFC7515], Section 5.1, Step 8 describes the method to create the final JWS Compact Serialization form of the PASSporT.

7. Compact Form of PASSporT

For a using protocol of PASSporT, the PASSporT claims as well as the PASSporT header may include redundant or default information that could be reconstructed at the destination based on information provided in the signaling protocol transporting the PASSporT object. In this case, it may be advantageous to have a more compact form of PASSporT to save the transmission of the bytes needed to represent the header and claims.

This specification defines the compact form of the PASSporT, in the spirit of the form defined in [RFC7515], Appendix F, with the use of two periods ("..") to represent the header and claim objects being removed, followed by the PASSporT signature as defined in Section 6, and the need for the destination to reconstruct the header and claim objects in order to verify the signature.

In order to construct the compact form of the PASSporT string, the procedure described in Section 6 MUST be used, with the exception of [RFC7515], Section 5.1, Step 8. This step would be replaced by the following construction of the compact form of PASSporT, ".." || BASE64URL(JWS Signature).

The using protocol of the compact form of PASSporT MUST be accompanied by a specification for how the header and claims objects can be reconstructed from information in the signaling protocol being used.

Note that the full form of the PASSporT, containing the entire header, payload, and signature, should also use the lexicographic ordering and whitespace serialization rules, particularly in the case where some using protocols or interworking between protocols may require switching between full and compact forms and maintaining the integrity of the signature.

7.1. Example Compact Form of PASSporT

The compact form of the following example token (with line breaks between periods used for readability purposes only)

```
eyJhbGciOiJFUzI1NiIsInR5cCI6InBhc3Nwb3J0IiwieDV1IjoiaHR0cHM6Ly9j
ZXJ0LmV4YW1wbGUub3JnL3Bhc3Nwb3J0LmNlciJ9
.
eyJkZXN0Ijp7InVyaSI6WyJzaXA6YWxpY2VAZXhhbXBsZS5jb20iXX0sImlhdCI
6IjE0NDMyMDgzNDUiLCJvcmlnIjp7InRuIjoiMTIxNTU1NTEyMTIifX0
.
rq3pjT1hoRwakEGjHCnWSwUnshd0-zJ6F1VOgFWSjHBr8Qjpjlk-cpFYpFYsojN
CpTzO3QfPOlckGaS6hEck7w
```

would be as follows:

```
..rq3pjT1hoRwakEGjHCnWSwUnshd0-zJ6F1VOgFWSjHBr8Qjpjlk-cpFYpFYsojN
CpTzO3QfPOlckGaS6hEck7w
```

8. Extending PASSporT

PASSporT includes the bare-minimum set of claims needed to securely assert the originating identity and support the secure properties discussed in various parts of this document. JWT supports a straightforward way to add additional asserted or signed information by simply adding new claims. PASSporT can be extended beyond the defined base set of claims to represent other information requiring assertion or validation beyond the originating identity itself as needed.

8.1. "ppt" (PASSporT) Header Parameter

Any using protocol can extend the payload of PASSporT with additional JWT claims. JWT claims are managed by the "JSON Web Token Claims" IANA registry as defined in [RFC7519], Section 10.1. Implementations of PASSporT MUST support the baseline claims defined in Section 5.2 and MAY support extended claims. If it is necessary for an extension to PASSporT to require that a relying party support a particular extended claim or set of claims in the PASSporT object, it can do so by specifying a "ppt" element for the PASSporT JOSE Header. All values of "ppt" need to be defined in a specification that associates the new value of the "ppt" element with the required claims and behaviors. Relying parties MUST fail to validate PASSporT objects containing an unsupported "ppt".

Using protocols MUST explicitly define how they carry each claim and the rules for how the header and payload objects are constructed beyond the lexicographical and serialization rules defined in this document.

Using protocols that carry the compact form of PASSporT (Section 7) instead of the full form MUST use only mandatory extensions signaled with "ppt" -- if a using protocol were to add additional optional claims to a PASSporT object it carried in compact form, relying parties would have no way to reconstruct the token. Moreover, using protocols that support the compact form of PASSporT MUST have some field to signal "ppt" to relying parties, as the compact form of PASSporT omits the JOSE Header.

8.2. Example Extended PASSporT Header

An example header with a PASSporT extension type of "foo" is as follows:

```
{
  "alg":"ES256",
  "ppt":"foo",
  "typ":"passport",
  "x5u":"https://tel.example.org/passport.cer"
}
```

8.3. Extended PASSporT Claims

Specifications that define extensions to the PASSporT mechanism MUST explicitly specify what claims they include beyond the base set of claims from this document, the order in which they will appear, and any further information necessary to implement the extension. All extensions MUST include the baseline PASSporT claim elements specified in Section 5; claims may only be appended to the claims object specified; they can never be removed or reordered. Specifying new claims follows the baseline JWT procedures ([RFC7519], Section 10.1). Understanding an extension or new claims defined by the extension on the destination verification of the PASSporT is optional. The creator of a PASSporT object cannot assume that destination systems will understand any given extension. Verification of PASSporTs by destination systems that do support an extension may then trigger appropriate application-level behavior in the presence of an extension; authors of extensions should provide appropriate extension-specific guidance to application developers on this point.

An example set of extended claims, extending the first example in Section 5.2.1.4 using "bar" as the newly defined claim, would be as follows:

```
{
  "bar":"beyond all recognition"
  "dest":{"uri":["sip:alice@example.com"]},
  "iat":1443208345,
  "orig":{"tn":"12155551212"}
}
```

9. Deterministic JSON Serialization

JSON objects can include spaces and line breaks, and key value pairs can occur in any order. It is therefore a non-deterministic string format. In order to make the digital signature verification work deterministically, the JSON representation of the JWS Protected Header object and JWS Payload object MUST be computed as follows.

The JSON object MUST follow the following rules. These rules are based on the thumbprint of a JSON Web Key (JWK) as defined in Section 3 Step 1 of [RFC7638].

1. The JSON object MUST contain no whitespace or line breaks before or after any syntactic elements.

2. JSON objects MUST have the keys ordered lexicographically by the Unicode [UNICODE] code points of the member names.

3. JSON value literals MUST be lowercase.

4. JSON numbers are to be encoded as integers unless the field is defined to be encoded otherwise.

5. Encoding rules MUST be applied recursively to member values and array values.

Note: For any PASSporT extension claims, member names within the scope of a JSON object MUST NOT be equal to other member names; otherwise, serialization will not be deterministic.

9.1. Example PASSporT Deterministic JSON Form

This section demonstrates the deterministic JSON serialization for the example PASSporT Payload shown in Section 5.2.1.4.

The initial JSON object is shown here:

```
{
  "dest":{"uri":["sip:alice@example.com"]},
  "orig":{"tn":"12155551212"}
  "iat":1443208345,
  "mky":[
    {
      "alg":"sha-256",
      "dig":"021ACC5427ABEB9C533F3E4B652E7D463F5442CD54
        F17A03A27DF9B07F4619B2"
    },
    {
      "alg":"sha-256",
      "dig":"4AADB9B13F82183B540212DF3E5D496B19E57C
        AB3E4B652E7D463F5442CD54F1"
    }
  ],
}
```

The parent members of the JSON object are as follows:

- "dest"
- "orig"
- "iat"
- "mky"

Their lexicographic order is:

- "dest"
- "iat"
- "mky"
- "orig"

The final constructed deterministic JSON serialization representation, with whitespace and line breaks removed (with line breaks used for display purposes only), is:

{"dest":{"uri":["sip:alice@example.com"],"iat":1443208345,"mky":
[{"alg":"sha-256","dig":"021ACC5427ABEB9C533F3E4B652E7D463F5442CD5
4F17A03A27DF9B07F4619B2"},{"alg":"sha-256","dig":"4AADB9B13F82183B5
40212DF3E5D496B19E57CAB3E4B652E7D463F5442CD54F1"}],
"orig":{"tn":"12155551212"}}

10. Security Considerations

10.1. Avoidance of Replay and Cut-and-Paste Attacks

There are a number of security considerations regarding the use of the token for the avoidance of replay and cut-and-paste attacks. PASSporTs SHOULD only be sent with application-level protocol information (e.g., for SIP, an INVITE as defined in [RFC3261]) corresponding to the required fields in the token. A unique set of token claims and token signature is constructed using the originating identity being asserted with the "orig" claim along with the following two claims:

- The "iat" claim should correspond to a date/time that the message was originated. It should also be within a relative time that is reasonable for clock drift and transmission time characteristics associated with the application using the PASSporT. Therefore, validation of the token should consider date and time correlation, which could be influenced by usage specific to the signaling protocol and by network time differences.

- The "dest" claim is included to further restrict the use of a valid PASSporT being sent as a replay attack to other destination parties. The verification of the PASSporT at the destination should verify that the "dest" claim matches the destination party as the intended recipient of the message.

10.2. Solution Considerations

The use of PASSporTs based on the validation of the digital signature and the associated certificate requires consideration of the authentication and authority or reputation of the signer to attest to the identity being asserted. The following considerations should be recognized when using PASSporT:

- The use of this token should not, in its own right, be considered a full solution for absolute non-repudiation of the identity being asserted.

- In many applications, the signer and the end user represented by the asserted identity may not be one and the same. For example, when a service provider signs and validates the token on behalf of the user consuming the service, the provider MUST have an authenticated and secure relationship with the end user or the device initiating and terminating the communications signaling.

- Applications that use PASSporT should ensure that the verification of the signature includes a means for verifying that the signer is authoritative through the use of an application-specific or service-specific set of common trust anchors for the application.

11. IANA Considerations

11.1. Media Type Registration

This section registers the "application/passport" media type (see [RFC2046] for the definition of "media type") in the "Media Types" registry in the manner described in [RFC6838], to indicate that the content is a PASSporT-defined JWT.

- Type name: application

- Subtype name: passport

- Required parameters: N/A

- Optional parameters: N/A

- Encoding considerations: 8bit; application/passport values are encoded as a series of base64url-encoded values (some of which may be the empty string) separated by period (".") characters.

- Security considerations: See the Security Considerations section of [RFC7515].

- Interoperability considerations: N/A
- Published specification: RFC 8225
- Applications that use this media type: Secure Telephone Identity Revisited (STIR) and other applications that require identity-related assertion
- Fragment identifier considerations: N/A
- Additional information:

  Magic number(s): N/A

  File extension(s): N/A

  Macintosh file type code(s): N/A

- Person & email address to contact for further information: Chris Wendt, chris-ietf@chriswendt.net
- Intended usage: COMMON
- Restrictions on usage: none
- Author: Chris Wendt <chris-ietf@chriswendt.net>
- Change Controller: IESG
- Provisional registration? No

11.2. Registrations in "JSON Web Token Claims"

Claim Name: "orig"
Claim Description: Originating Identity String
Change Controller: IESG
Reference: Section 5.2.1 of RFC 8225

Claim Name: "dest"
Claim Description: Destination Identity String
Change Controller: IESG
Reference: Section 5.2.1 of RFC 8225

Claim Name: "mky"
Claim Description: Media Key Fingerprint String
Change Controller: IESG
Reference: Section 5.2.2 of RFC 8225

11.3. Registration in "JSON Web Signature and Encryption Header Parameters"

Header Parameter Name: "ppt"
Header Parameter Description: PASSporT extension identifier
Header Parameter Usage Location(s): JWS
Change Controller: IESG
Reference: Section 8.1 of RFC 8225

11.4. PASSporT Extensions Registry

The IANA has created a new PASSporT Type registry for "ppt" parameter values. That parameter and its values are defined in Section 8.1. New registry entries must contain the name of the "ppt" parameter value and the specification in which the value is described. The policy for this registry is Specification Required [RFC8126].

12. References

12.1. Normative References

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <https://www.rfc-editor.org/info/rfc4566>.

[RFC4572] Lennox, J., "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 4572, DOI 10.17487/RFC4572, July 2006, <https://www.rfc-editor.org/info/rfc4572>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <https://www.rfc-editor.org/info/rfc6979>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC7638] Jones, M. and N. Sakimura, "JSON Web Key (JWK) Thumbprint", RFC 7638, DOI 10.17487/RFC7638, September 2015, <https://www.rfc-editor.org/info/rfc7638>.

[RFC8122] Lennox, J. and C. Holmberg, "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 8122, DOI 10.17487/RFC8122, March 2017, <https://www.rfc-editor.org/info/rfc8122>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8224] Peterson, J., Jennings, C., Rescorla, E., and C. Wendt, "Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 8224, DOI 10.17487/RFC8224, February 2018, <https://www.rfc-editor.org/info/rfc8224>.

[UNICODE] The Unicode Consortium, "The Unicode Standard", <http://www.unicode.org/versions/latest/>.

12.2. Informative References

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC7340] Peterson, J., Schulzrinne, H., and H. Tschofenig, "Secure Telephone Identity Problem Statement and Requirements", RFC 7340, DOI 10.17487/RFC7340, September 2014, <https://www.rfc-editor.org/info/rfc7340>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

Appendix A. Example ES256-Based PASSporT JWS Serialization and Signature

For PASSporT, there will always be a JWS with the following members:

- "protected", with the value BASE64URL(UTF8(JWS Protected Header))
- "payload", with the value BASE64URL(JWS Payload)
- "signature", with the value BASE64URL(JWS Signature)

This example will follow the steps in JWS ([RFC7515], Section 5.1, Steps 1-6 and 8); it incorporates the additional serialization steps required for PASSporT.

Step 1 for JWS references the JWS Payload. An example PASSporT Payload is as follows:

```
{
  "dest":{"uri":["sip:alice@example.com"]}
  "iat":1471375418,
  "orig":{"tn":"12155551212"}
}
```

This would be serialized to the following form (with line break used for display purposes only):

```
{"dest":{"uri":["sip:alice@example.com"]},"iat":1471375418,
"orig":{"tn":"12155551212"}}
```

Step 2 computes the BASE64URL(JWS Payload), producing this value (with line break used for display purposes only):

```
eyJkZXN0Ijp7InVyaSI6WyJzaXA6YWxpY2VAZXhhbXBsZS5jb20iXX0sImlhdCI
6MTQ3MTM3NTQxOCwib3JpZyI6eyJ0biI6IjEyMTU1NTUxMjEyIn19
```

For Step 3, an example PASSporT Protected Header constructed as a JOSE Header is as follows:

```
{
  "alg":"ES256",
  "typ":"passport",
  "x5u":"https://cert.example.org/passport.cer"
}
```

This would be serialized to the following form (with line break used for display purposes only):

```
{"alg":"ES256","typ":"passport","x5u":"https://cert.example.org
  /passport.cer"}
```

Step 4 performs the BASE64URL(UTF8(JWS Protected Header)) operation and encoding, producing this value (with line break used for display purposes only):

```
eyJhbGciOiJFUzI1NiIsInR5cCI6InBhc3Nwb3J0IiwieDV1IjoiaHR0cHM6Ly9j
ZXJ0LmV4YW1wbGUub3JnL3Bhc3Nwb3J0LmNlciJ9
```

Steps 5 and 6 perform the computation of the digital signature of the PASSporT Signing Input ASCII(BASE64URL(UTF8(JWS Protected Header)) || "." || BASE64URL(JWS Payload)), using ES256 as the algorithm and the BASE64URL(JWS Signature).

```
VLBCIVDCaeK6M4hLJb6SHQvacAQVvoiiEOWQ_iUkqk79UD81fHQ0E1b3_GluIkb
a7UWYRM47ZbNFdOJquE35cw
```

Step 8 describes how to create the final PASSporT, concatenating the values in the order Header.Payload.Signature with period (".") characters. For the above example values, this would produce the following (with line breaks between periods used for readability purposes only):

```
eyJhbGciOiJFUzI1NiIsInR5cCI6InBhc3Nwb3J0IiwieDV1IjoiaHR0cHM6Ly9j
ZXJ0LmV4YW1wbGUub3JnL3Bhc3Nwb3J0LmNlciJ9
.
eyJkZXN0Ijp7InVyaSI6WyJzaXA6YWxpY2VAZXhhbXBsZS5jb20iXX0sImlhdCI
6MTQ3MTM3NTQxOCwib3JpZyI6eyJ0biI6IjEyMTU1NTUxMjEyIn19
.
VLBCIVDCaeK6M4hLJb6SHQvacAQVvoiiEOWQ_iUkqk79UD81fHQ0E1b3_GluIkb
a7UWYRM47ZbNFdOJquE35cw
```

A.1. X.509 Private Key in PKCS #8 Format for ES256 Example

```
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgi7q2TZvN9VDFg8Vy
qCP06bETrR2v8MRvr89rn4i+UAahRANCAAQWfaj1HUETpoNCrOtp9KA8o0V79IuW
ARKt9C1cFPkyd3FBP4SeiNZxQhDrD0tdBHls3/wFe8++K2FrPyQF9vuh
-----END PRIVATE KEY-----
```

A.2. X.509 Public Key for ES256 Example

```
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE8HNbQd/TmvCKwPKHkMF9fScavGeH
78YTU8qLS8I5HLHSSmlATLcslQMhNC/OhlWBYC626nIlo7XeebYS7Sb37g==
-----END PUBLIC KEY-----
```

Acknowledgments

Particular thanks to members of the ATIS and SIP Forum NNI Task Group, including Jim McEachern, Martin Dolly, Richard Shockey, John Barnhill, Christer Holmberg, Victor Pascual Avila, Mary Barnes, and Eric Burger, for their review, ideas, and contributions. Thanks also to Henning Schulzrinne, Russ Housley, Alan Johnston, Richard Barnes, Mark Miller, Ted Hardie, Dave Crocker, Robert Sparks, and Jim Schaad for valuable feedback on the technical and security aspects of the document. Additional thanks to Harsha Bellur for assistance in coding the example tokens.

Authors' Addresses

Chris Wendt
Comcast
One Comcast Center
Philadelphia, PA 19103
United States of America

Email: chris-ietf@chriswendt.net

Jon Peterson
Neustar Inc.
1800 Sutter St. Suite 570
Concord, CA 94520
United States of America

Email: jon.peterson@neustar.biz