

OAuth Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: June 18, 2016

M. Jones  
Microsoft  
J. Bradley  
Ping Identity  
H. Tschofenig  
ARM Limited  
December 16, 2015

**Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)**  
**draft-ietf-oauth-proof-of-possession-10**

Abstract

This specification defines how to declare in a JSON Web Token (JWT) that the presenter of the JWT possesses a particular proof-of-possession key and that the recipient can cryptographically confirm proof-of-possession of the key by the presenter. Being able to prove possession of a key is also sometimes described as the presenter being a holder-of-key.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 18, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                             |  |                    |
|-----------------------------|--|--------------------|
| <a href="#">1.</a>          | <a href="#">Introduction</a>   | <a href="#">3</a>  |
| <a href="#">1.1.</a>        | <a href="#">Notational Conventions</a>   | <a href="#">5</a>  |
| <a href="#">2.</a>          | <a href="#">Terminology</a>  | <a href="#">5</a>  |
| <a href="#">3.</a>          | <a href="#">Representations for Proof-of-Possession Keys</a>                     | <a href="#">6</a>  |
| <a href="#">3.1.</a>        | <a href="#">Confirmation Claim</a>   | <a href="#">6</a>  |
| <a href="#">3.2.</a>        | <a href="#">Representation of an Asymmetric Proof-of-Possession Key</a>          | <a href="#">7</a>  |
| <a href="#">3.3.</a>        | <a href="#">Representation of an Encrypted Symmetric Proof-of-Possession Key</a> | <a href="#">8</a>  |
| <a href="#">3.4.</a>        | <a href="#">Representation of a Key ID for a Proof-of-Possession Key</a>         | <a href="#">9</a>  |
| <a href="#">3.5.</a>        | <a href="#">Representation of a URL for a Proof-of-Possession Key</a>            | <a href="#">9</a>  |
| <a href="#">3.6.</a>        | <a href="#">Specifics Intentionally Not Specified</a>                            | <a href="#">10</a> |
| <a href="#">4.</a>          | <a href="#">Security Considerations</a>  | <a href="#">10</a> |
| <a href="#">5.</a>          | <a href="#">Privacy Considerations</a>   | <a href="#">11</a> |
| <a href="#">6.</a>          | <a href="#">IANA Considerations</a>  | <a href="#">11</a> |
| <a href="#">6.1.</a>        | <a href="#">JSON Web Token Claims Registration</a>                               | <a href="#">12</a> |
| <a href="#">6.1.1.</a>      | <a href="#">Registry Contents</a>  | <a href="#">12</a> |
| <a href="#">6.2.</a>        | <a href="#">JWT Confirmation Methods Registry</a>                                | <a href="#">12</a> |
| <a href="#">6.2.1.</a>      | <a href="#">Registration Template</a>  | <a href="#">12</a> |
| <a href="#">6.2.2.</a>      | <a href="#">Initial Registry Contents</a>  | <a href="#">13</a> |
| <a href="#">7.</a>          | <a href="#">References</a>   | <a href="#">13</a> |
| <a href="#">7.1.</a>        | <a href="#">Normative References</a>   | <a href="#">13</a> |
| <a href="#">7.2.</a>        | <a href="#">Informative References</a>   | <a href="#">14</a> |
| <a href="#">Appendix A.</a> | <a href="#">Acknowledgements</a>   | <a href="#">15</a> |
| <a href="#">Appendix B.</a> | <a href="#">Document History</a>   | <a href="#">15</a> |
|                             | <a href="#">Authors' Addresses</a>   | <a href="#">17</a> |

## 1. Introduction

This specification defines how a JSON Web Token [JWT] can declare that the presenter of the JWT possesses a particular proof-of-possession key and that the recipient can cryptographically confirm proof-of-possession of the key by the presenter. Proof-of-possession of a key is also sometimes described as the presenter being a holder-of-key. The [I-D.ietf-oauth-pop-architecture] specification describes key confirmation, among other confirmation mechanisms. This specification defines how to communicate key confirmation key information in JWTs.

Envision the following two use cases. The first use case employs a symmetric proof-of-possession key and the second use case employs an asymmetric proof-of-possession key.

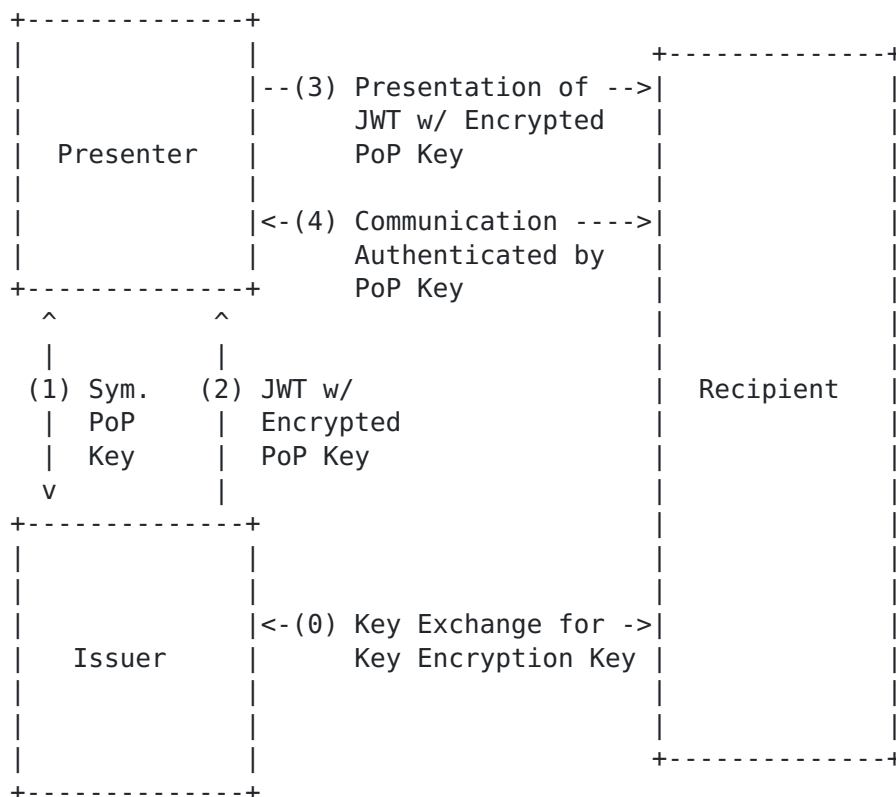


Figure 1: Proof-of-Possession with a Symmetric Key

In the case illustrated in Figure 1, either the presenter generates a symmetric key and privately sends it to the issuer (1) or the issuer generates a symmetric key and privately sends it to the presenter (1). The issuer generates a JWT with an encrypted copy of this symmetric key in the confirmation claim. This symmetric key is



encrypted with a key known only to the issuer and the recipient, which was previously established in step (0). The entire JWT is integrity protected by the issuer. The JWT is then (2) sent to the presenter. Now, the presenter is in possession of the symmetric key as well as the JWT (which includes the confirmation claim). When the presenter (3) presents the JWT to the recipient, it also needs to demonstrate possession of the symmetric key; the presenter, for example, (4) uses the symmetric key in a challenge/response protocol with the recipient. The recipient is then able to verify that it is interacting with the genuine presenter by decrypting the key in the confirmation claim of the JWT. By doing this, the recipient obtains the symmetric key, which it then uses to verify cryptographically protected messages exchanged with the presenter (4). This symmetric key mechanism described above is conceptually similar to the use of Kerberos tickets.

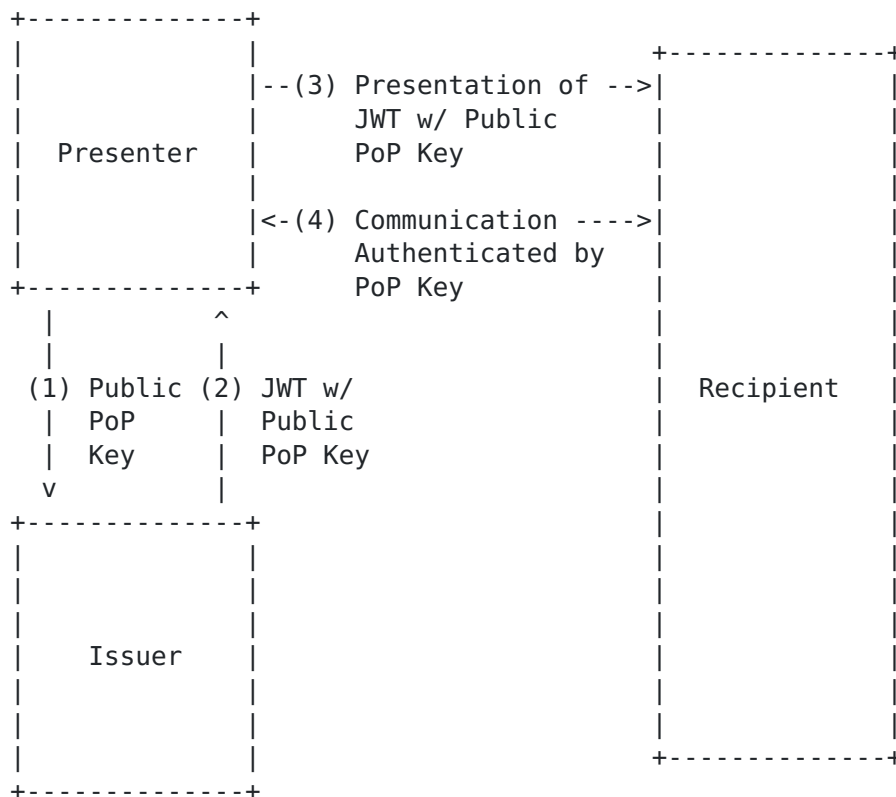


Figure 2: Proof-of-Possession with an Asymmetric Key

In the case illustrated in Figure 2, the presenter generates a public/private key pair and (1) sends the public key to the issuer, which creates a JWT that contains the public key (or an identifier for it) in the confirmation claim. The entire JWT is integrity protected using a digital signature to protect it against



modifications. The JWT is then (2) sent to the presenter. When the presenter (3) presents the JWT to the recipient, it also needs to demonstrate possession of the private key. The presenter, for example, (4) uses the private key in a TLS exchange with the recipient or (4) signs a nonce with the private key. The recipient is able to verify that it is interacting with the genuine presenter by extracting the public key from the confirmation claim of the JWT (after verifying the digital signature of the JWT) and utilizing it with the private key in the TLS exchange or by checking the nonce signature.

In both cases, the JWT may contain other claims that are needed by the application.

### **1.1. Notational Conventions**

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 [RFC 2119](#) [RFC2119].

Unless otherwise noted, all the protocol parameter names and values are case sensitive.

## **2. Terminology**

This specification uses terms defined in the JSON Web Token [JWT], JSON Web Key [JWK], and JSON Web Encryption [JWE] specifications.

These terms are defined by this specification:

#### **Issuer**

Party that creates the JWT and binds the proof-of-possession key to it.

#### **Presenter**

Party that proves possession of a private key (for asymmetric key cryptography) or secret key (for symmetric key cryptography) to a recipient.

#### **Recipient**

Party that receives the JWT containing the proof-of-possession key information from the presenter.





### 3. Representations for Proof-of-Possession Keys

By including a "cnf" (confirmation) claim in a JWT, the issuer of the JWT declares that the presenter possesses a particular key, and that the recipient can cryptographically confirm that the presenter has possession of that key. The value of the "cnf" claim is a JSON object and the members of that object identify the proof-of-possession key.

The presenter can be identified in one of several ways by the JWT, depending upon the application requirements. If the JWT contains a "sub" (subject) claim [[JWT](#)], the presenter is normally the subject identified by the JWT. (In some applications, the subject identifier will be relative to the issuer identified by the "iss" (issuer) claim [[JWT](#)].) If the JWT contains no "sub" (subject) claim, the presenter is normally the issuer identified by the JWT using the "iss" (issuer) claim. The case in which the presenter is the subject of the JWT is analogous to SAML 2.0 [[OASIS.saml-core-2.0-os](#)] SubjectConfirmation usage. At least one of the "sub" and "iss" claims MUST be present in the JWT. Some use cases may require that both be present.

Another means used by some applications to identify the presenter is an explicit claim, such as the "azp" (authorized party) claim defined by OpenID Connect [[OpenID.Core](#)]. Ultimately, the means of identifying the presenter is application-specific, as is the means of confirming possession of the key that is communicated.

#### 3.1. Confirmation Claim

The "cnf" (confirmation) claim is used in the JWT to contain members used to identify the proof-of-possession key. Other members of the "cnf" object may be defined because a proof-of-possession key may not be the only means of confirming the authenticity of the token. This is analogous to the SAML 2.0 [[OASIS.saml-core-2.0-os](#)] SubjectConfirmation element, in which a number of different subject confirmation methods can be included, including proof-of-possession key information.

The set of confirmation members that a JWT must contain to be considered valid is context dependent and is outside the scope of this specification. Specific applications of JWTs will require implementations to understand and process some confirmation members in particular ways. However, in the absence of such requirements, all confirmation members that are not understood by implementations MUST be ignored.

This specification establishes the IANA "JWT Confirmation Methods" registry for these members in [Section 6.2](#) and registers the members



defined by this specification. Other specifications can register other members used for confirmation, including other members for conveying proof-of-possession keys, possibly using different key representations.

The "cnf" claim value MUST represent only a single proof-of-possession key; thus, at most one of the "jwk", "jwe", and "jku" confirmation values defined below may be present. Note that if an application needs to represent multiple proof-of-possession keys in the same JWT, one way for it to achieve this is to use other claim names, in addition to "cnf", to hold the additional proof-of-possession key information. These claims could use the same syntax and semantics as the "cnf" claim. Those claims would be defined by applications or other specifications and could be registered in the IANA "JSON Web Token Claims" registry [[IANA.JWT.Claims](#)].

### 3.2. Representation of an Asymmetric Proof-of-Possession Key

When the key held by the presenter is an asymmetric private key, the "jwk" member is a JSON Web Key [[JWK](#)] representing the corresponding asymmetric public key. The following example demonstrates such a declaration in the JWT Claims Set of a JWT:

```
{
  "iss": "https://server.example.com",
  "aud": "https://client.example.org",
  "exp": 1361398824,
  "cnf": {
    "jwk": {
      "kty": "EC",
      "use": "sig",
      "crv": "P-256",
      "x": "18wHLeIgW9wVN6VD1Txgpqy2LszYkMf6J8njVAibvhM",
      "y": "-V4dS4UaLMgP_4fY4j8ir7cl1TXlFdAgcx55o7TkCSA"
    }
  }
}
```

The JWK MUST contain the required key members for a JWK of that key type and MAY contain other JWK members, including the "kid" (key ID) member.

The "jwk" member MAY also be used for a JWK representing a symmetric key, provided that the JWT is encrypted so that the key is not revealed to unintended parties. If the JWT is not encrypted, the symmetric key MUST be encrypted as described below.



### 3.3. Representation of an Encrypted Symmetric Proof-of-Possession Key

When the key held by the presenter is a symmetric key, the "jwe" member is an encrypted JSON Web Key [[JWK](#)] encrypted to a key known to the recipient using the JWE Compact Serialization containing the symmetric key. The rules for encrypting a JWK are found in [Section 7](#) of the JSON Web Key [[JWK](#)] specification.

The following example illustrates a symmetric key that could subsequently be encrypted for use in the "jwe" member:

```
{
  "kty": "oct",
  "alg": "HS256",
  "k": "ZoRS0rFzN_FzUA5XKMYoVHyzff5oRJxl-IXRtztJ6uE"
}
```

The UTF-8 [[RFC3629](#)] encoding of this JWK is used as the JWE Plaintext when encrypting the key.

The following example is a JWE Header that could be used when encrypting this key:

```
{
  "alg": "RSA-OAEP",
  "enc": "A128CBC-HS256"
}
```

The following example JWT Claims Set of a JWT illustrates the use of an encrypted symmetric key as the "jwe" member value:

```
{
  "iss": "https://server.example.com",
  "sub": "24400320",
  "aud": "s6BhdRkqt3",
  "nonce": "n-0S6_WzA2Mj",
  "exp": 1311281970,
  "iat": 1311280970,
  "cnf": {
    "jwe":
      "eyJhbGciOiJSU0EtT0FFUCIsImVuYyI6IkExMjhdQkMtSFMyNTYifQ.
      (remainder of JWE omitted for brevity)"
  }
}
```



### **3.4. Representation of a Key ID for a Proof-of-Possession Key**

The proof-of-possession key can also be identified by the use of a Key ID instead of communicating the actual key, provided the recipient is able to obtain the identified key using the Key ID. In this case, the issuer of a JWT declares that the presenter possesses a particular key and that the recipient can cryptographically confirm proof-of-possession of the key by the presenter by including a "cnf" (confirmation) claim in the JWT whose value is a JSON object, with the JSON object containing a "kid" (key ID) member identifying the key.

The following example demonstrates such a declaration in the JWT Claims Set of a JWT:

```
{
  "iss": "https://server.example.com",
  "aud": "https://client.example.org",
  "exp": 1361398824,
  "cnf": {
    "kid": "dfd1aa97-6d8d-4575-a0fe-34b96de2bfad"
  }
}
```

The content of the "kid" value is application specific. For instance, some applications may choose to use a JWK Thumbprint [[JWK.Thumbprint](#)] value as the "kid" value.

### **3.5. Representation of a URL for a Proof-of-Possession Key**

The proof-of-possession key can be passed by reference instead of being passed by value. This is done using the "jku" (JWK Set URL) member. Its value is a URI [[RFC3986](#)] that refers to a resource for a set of JSON-encoded public keys represented as a JWK Set [[JWK](#)], one of which is the proof-of-possession key. If there are multiple keys in the referenced JWK Set document, a "kid" member MUST also be included, with the referenced key's JWK also containing the same "kid" value.

The protocol used to acquire the resource MUST provide integrity protection. An HTTP GET request to retrieve the JWK Set MUST use Transport Layer Security (TLS) [[RFC5246](#)] and the identity of the server MUST be validated, as per [Section 6 of RFC 6125](#) [[RFC6125](#)].

The following example demonstrates such a declaration in the JWT Claims Set of a JWT:





```
{
  "iss": "https://server.example.com",
  "sub": "17760704",
  "aud": "https://client.example.org",
  "exp": 1440804813,
  "cnf": {
    "jku": "https://keys.example.net/pop-keys.json",
    "kid": "2015-08-28"
  }
}
```

### **3.6. Specifics Intentionally Not Specified**

Proof-of-possession is typically demonstrated by having the presenter sign a value determined by the recipient using the key possessed by the presenter. This value is sometimes called a "nonce" or a "challenge".

The means of communicating the nonce and the nature of its contents are intentionally not described in this specification, as different protocols will communicate this information in different ways. Likewise, the means of communicating the signed nonce is also not specified, as this is also protocol-specific.

Note that another means of proving possession of the key when it is a symmetric key is to encrypt the key to the recipient. The means of obtaining a key for the recipient is likewise protocol-specific.

For examples using the mechanisms defined in this specification, see [[I-D.ietf-oauth-pop-architecture](#)].

## **4. Security Considerations**

All of the security considerations that are discussed in [[JWT](#)] also apply here. In addition, proof-of-possession introduces its own unique security issues. Possessing a key is only valuable if it is kept secret. Appropriate means must be used to ensure that unintended parties do not learn private key or symmetric key values.

Proof-of-possession via encrypted symmetric secrets is subject to replay attacks. This attack can be avoided when a signed nonce or challenge is used, since the recipient can use a distinct nonce or challenge for each interaction.

Similarly to other information included in a JWT, it is necessary to apply data origin authentication and integrity protection (via a keyed message digest or a digital signature). Data origin



authentication ensures that the recipient of the JWT learns about the entity that created the JWT, since this will be important for any policy decisions. Integrity protection prevents an adversary from changing any elements conveyed within the JWT payload. Special care has to be applied when carrying symmetric keys inside the JWT, since those not only require integrity protection, but also confidentiality protection.

A recipient might not understand the "cnf" claim. Applications that require the proof-of-possession keys communicated with it to be understood and processed must ensure that the parts of this specification that they use are implemented.

Applications utilizing proof-of-possession should also utilize audience restriction, as described in Section 4.1.3 of [JWT], as it provides different protections. Proof-of-possession can be used by recipients to reject messages from unauthorized senders. Audience restriction can be used by recipients to reject messages intended for different recipients.

## 5. Privacy Considerations

A proof-of-possession key can be used as a correlation handle if the same key is used with multiple parties. Thus, for privacy reasons, it is recommended that different proof-of-possession keys be used when interacting with different parties.

## 6. IANA Considerations

The following registration procedure is used for all the registries established by this specification.

Values are registered on a Specification Required [RFC5226] basis after a three-week review period on the `oauth-pop-reg-review@ietf.org` mailing list, on the advice of one or more Designated Experts. However, to allow for the allocation of values prior to publication, the Designated Experts may approve registration once they are satisfied that such a specification will be published. [[ Note to the RFC Editor: The name of the mailing list should be determined in consultation with the IESG and IANA. Suggested name: `oauth-pop-reg-review@ietf.org`. ]]

Registration requests sent to the mailing list for review should use an appropriate subject (e.g., "Request to register JWT Confirmation Method: example"). Registration requests that are undetermined for a period longer than 21 days can be brought to the IESG's attention



(using the [iesg@ietf.org](mailto:iesg@ietf.org) mailing list) for resolution.

Criteria that should be applied by the Designated Experts include determining whether the proposed registration duplicates existing functionality, determining whether it is likely to be of general applicability or whether it is useful only for a single application, and whether the registration makes sense.

It is suggested that multiple Designated Experts be appointed who are able to represent the perspectives of different applications using this specification, in order to enable broadly-informed review of registration decisions. In cases where a registration decision could be perceived as creating a conflict of interest for a particular Expert, that Expert should defer to the judgment of the other Experts.

## **6.1. JSON Web Token Claims Registration**

This specification registers the "cnf" claim in the IANA "JSON Web Token Claims" registry [[IANA.JWT.Claims](#)] established by [[JWT](#)].

### **6.1.1. Registry Contents**

- o Claim Name: "cnf"
- o Claim Description: Confirmation
- o Change Controller: IESG
- o Specification Document(s): [Section 3.1](#) of [[ this document ]]

## **6.2. JWT Confirmation Methods Registry**

This specification establishes the IANA "JWT Confirmation Methods" registry for JWT "cnf" member values. The registry records the confirmation method member and a reference to the specification that defines it.

### **6.2.1. Registration Template**

Confirmation Method Value:

The name requested (e.g., "kid"). Because a core goal of this specification is for the resulting representations to be compact, it is RECOMMENDED that the name be short -- not to exceed 8 characters without a compelling reason to do so. This name is case-sensitive. Names may not match other registered names in a case-insensitive manner unless the Designated Experts state that there is a compelling reason to allow an exception.



**Confirmation Method Description:**

Brief description of the confirmation method (e.g., "Key Identifier").

**Change Controller:**

For Standards Track RFCs, list the "IESG". For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

**Specification Document(s):**

Reference to the document or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. An indication of the relevant sections may also be included but is not required.

**6.2.2. Initial Registry Contents**

- o Confirmation Method Value: "jwk"
- o Confirmation Method Description: JSON Web Key Representing Public Key
- o Change Controller: IESG
- o Specification Document(s): [Section 3.2](#) of [[ this document ]]
  
- o Confirmation Method Value: "jwe"
- o Confirmation Method Description: Encrypted JSON Web Key
- o Change Controller: IESG
- o Specification Document(s): [Section 3.3](#) of [[ this document ]]
  
- o Confirmation Method Value: "kid"
- o Confirmation Method Description: Key Identifier
- o Change Controller: IESG
- o Specification Document(s): [Section 3.4](#) of [[ this document ]]
  
- o Confirmation Method Value: "jku"
- o Confirmation Method Description: JWK Set URL
- o Change Controller: IESG
- o Specification Document(s): [Section 3.5](#) of [[ this document ]]

**7. References****7.1. Normative References**

[IANA.JWT.Claims]

IANA, "JSON Web Token Claims",  
<<http://www.iana.org/assignments/jwt>>.

[JWE]

Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)",





- [RFC 7516](#), DOI 10.17487/RFC7156, May 2015, <<http://www.rfc-editor.org/info/rfc7516>>.
- [JWK] Jones, M., "JSON Web Key (JWK)", [RFC 7517](#), DOI 10.17487/RFC7157, May 2015, <<http://www.rfc-editor.org/info/rfc7517>>.
- [JWT] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", [RFC 7519](#), DOI 10.17487/RFC7159, May 2015, <<http://www.rfc-editor.org/info/rfc7519>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc3986>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 5226](#), DOI 10.17487/RFC5226, May 2008, <<http://www.rfc-editor.org/info/rfc5226>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", [RFC 6125](#), DOI 10.17487/RFC6125, March 2011, <<http://www.rfc-editor.org/info/rfc6125>>.

## **7.2. Informative References**

- [I-D.ietf-oauth-pop-architecture]  
Hunt, P., Richer, J., Mills, W., Mishra, P., and H. Tschofenig, "OAuth 2.0 Proof-of-Possession (PoP) Security Architecture", [draft-ietf-oauth-pop-architecture-05](#) (work



in progress), October 2015.

[JWK.Thumbprint]

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

[OASIS.saml-core-2.0-os]

Cantor, S., Kemp, J., Philpott, R., and E. Maler, "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard saml-core-2.0-os, March 2005.

[OpenID.Core]

Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0", November 2014, <[http://openid.net/specs/openid-connect-core-1\\_0.html](http://openid.net/specs/openid-connect-core-1_0.html)>.

## **Appendix A. Acknowledgements**

The authors wish to thank Brian Campbell, Kepeng Li, James Manger, Kathleen Moriarty, Justin Richer, and Nat Sakimura for their reviews of the specification.

## **Appendix B. Document History**

[[ to be removed by the RFC Editor before publication as an RFC ]]

-10

- o Addressed review comments by Barry Leiba.

-09

- o Removed erroneous quotation marks around numeric "exp" claim values in examples.

-08

- o Added security consideration about also utilizing audience restriction.

-07

- o Addressed review comments by Hannes Tschofenig, Kathleen Moriarty, and Justin Richer. Changes were:

- o Clarified that symmetric proof-of-possession keys can be generated by either the presenter or the issuer.
- o Clarified that confirmation members that are not understood must be ignored unless otherwise specified by the application.

-06

- o Added diagrams to the introduction.

-05

- o Addressed review comments by Kepeng Li.

-04

- o Allowed the use of "jwk" for symmetric keys when the JWT is encrypted.
- o Added the "jku" (JWK Set URL) member.
- o Added privacy considerations.
- o Reordered sections so that the "cnf" (confirmation) claim is defined before it is used.
- o Noted that applications can define new claim names, in addition to "cnf", to represent additional proof-of-possession keys, using the same representation as "cnf".
- o Applied wording clarifications suggested by Nat Sakimura.

-03

- o Separated the "jwk" and "jwe" confirmation members; the former represents a public key as a JWK and the latter represents a symmetric key as a JWE encrypted JWK.
- o Changed the title to indicate that a proof-of-possession key is being communicated.
- o Updated language that formerly assumed that the issuer was an OAuth 2.0 authorization server.
- o Described ways that applications can choose to identify the presenter, including use of the "iss", "sub", and "azp" claims.

- o Harmonized the registry language with that used in JWT [[RFC 7519](#)].
- o Addressed other issues identified during working group last call.
- o Referenced the JWT and JOSE RFCs.

-02

- o Defined the terms Issuer, Presenter, and Recipient and updated their usage within the document.
- o Added a description of a use case using an asymmetric proof-of-possession key to the introduction.
- o Added the "kid" (key ID) confirmation method.
- o These changes address the open issues identified in the previous draft.

-01

- o Updated references.

-00

- o Created the initial working group draft from [draft-jones-oauth-proof-of-possession-02](#).

#### Authors' Addresses

Michael B. Jones  
Microsoft

Email: [mbj@microsoft.com](mailto:mbj@microsoft.com)

URI: <http://self-issued.info/>

John Bradley  
Ping Identity

Email: [ve7jtb@ve7jtb.com](mailto:ve7jtb@ve7jtb.com)

URI: <http://www.thread-safe.com/>

Hannes Tschofenig  
ARM Limited  
Austria

Email: Hannes.Tschofenig@gmx.net  
URI: <http://www.tschofenig.priv.at>