

Network Working Group	D. Smith	
Internet-Draft	Individual	
Intended status: Informational	P. Motykowski	
Expires: February 13, 2009	Y. Faruqi	
	PingID	
	P. Saint-Andre, Ed.	
	Individual	
	August 12, 2008	

[TOC](#)

**OpenToken
draft-smith-opentoken-02**

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on February 13, 2009.

Abstract

This document describes OpenToken (OTK), a format for the lightweight, secure, cross-application exchange of key-value pairs. The format is designed primarily for use as an HTTP cookie or query parameter, but can also be used in other scenarios that require a compact, application-neutral token.

Table of Contents

- [1. Introduction](#)
 - [1.1. Motivation](#)
 - [1.2. Terminology](#)
- [2. Token Layout](#)
- [3. Processing Rules](#)
 - [3.1. Encoding](#)
 - [3.2. Decoding](#)
 - [3.3. Standard Key-Value Pairs](#)
- [4. Cipher Suites](#)
- [5. Payload Format](#)
- [6. Canonical Test Data](#)
 - [6.1. Test Case 1: AES-128](#)
 - [6.2. Test Case 2: AES-256](#)
 - [6.3. Test Case 3: 3DES-168](#)
- [7. Security Considerations](#)
- [8. References](#)
 - [8.1. Normative References](#)
 - [8.2. Informative References](#)
- [§ Authors' Addresses](#)
- [§ Intellectual Property and Copyright Statements](#)

1. Introduction

[TOC](#)

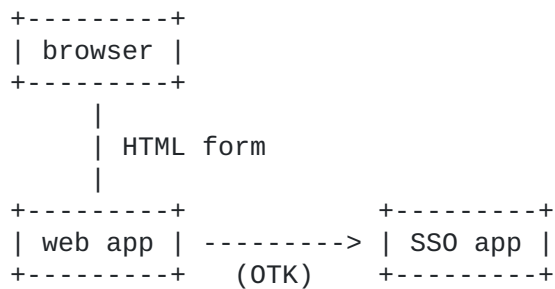
1.1. Motivation

[TOC](#)

This document describes OpenToken (OTK), a format for the lightweight, secure, cross-application exchange of key-value pairs between applications that use HTTP (see [\[RFC2616\]](#) ([Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," June 1999.](#))) as the transport protocol. The format is designed primarily for use as an HTTP cookie (see [\[RFC2965\]](#) ([Kristol, D. and L. Montulli, "HTTP State Management Mechanism," October 2000.](#))) or query parameter, but can also be used in other scenarios that require a compact, application-neutral token.

The OpenToken technology is not designed to encapsulate formal identity assertions (for which see [\[SAML\]](#) ([Cantor, S., Kemp, J., Philpott, R., and E. Maler, "Assertions and Protocol for the OASIS Security Assertion Markup Language \(SAML\) V2.0," March 2005.](#))) or authentication credentials (for which see [\[SASL\]](#) ([Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer \(SASL\)," June 2006.](#))). Instead, OpenToken is designed to encapsulate basic name-value pairs for exchange between applications that use HTTP as the transport protocol.

Consider the example of a web application that receives information from an end user via a web browser and subsequently shares that information with a backend system such as a single-sign-on (SSO) application.



Naturally the web application or the single-sign-on application could exchange the same information with other applications (e.g., billing, customer service, enterprise resource planning) or push the information back to the end user via an HTTP cookie. The end user could also share that same information with other web applications (e.g., the web application could store the information on the end user's browser via an HTTP cookie, which could be shared with other applications).

The use of a consistent data format enables a more seamless exchange of information between applications (e.g., by removing the need to translate between different application-specific data formats).

1.2. Terminology

[TOC](#)

The following keywords are to be interpreted as described in [\[RFC2119\] \(Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.\)](#): "MUST", "SHALL", "REQUIRED"; "MUST NOT", "SHALL NOT"; "SHOULD", "RECOMMENDED"; "SHOULD NOT", "NOT RECOMMENDED"; "MAY", "OPTIONAL".

2. Token Layout

[TOC](#)

The OpenToken format is specified in the following table.

Byte Range	Length	Description
0..2	3	'O','T','K' literal
3	1	Version identifier
4	1	Cipher suite identifier
5..24	20	SHA-1 HMAC
25	1	IV length
26..x	x-26	IV
x+1	1	Key Info length
x+2..y	y-(x+2)	Key info
y+1..y+3	2	Payload length
y+4..z	z-(y+4)	Payload
TOTAL	29+x+y+z	N/A

The following notes apply to the foregoing token parameters:

*The datatype for the version identifier, cipher suite identifier, IV length, and Key Info length is unsigned byte.

*The initialization vector (IV) has a maximum length of 255 bytes. This field is OPTIONAL and MAY have a length of 0 (IV length) to indicate that no IV is available for this token. For details about initialization vectors, see [\[RFC2898\] \(Kaliski, B., "PKCS #5: Password-Based Cryptography Specification Version 2.0," September 2000.\)](#).

*The payload is a series of key-value pairs, as described under [Section 5 \(Payload Format\)](#).

*The payload has a maximum (compressed) length of 64k bytes. While this format supports a payload of 64k bytes, deployment scenarios that pass the token using HTTP (either as a query parameter or cookie) SHOULD limit the token length to less than 4k for optimal compatibility.

*The [\[HMAC\] \(National Institute of Standards and Technology, "The Keyed-Hash Message Authentication Code \(HMAC\)," March 2002.\)](#) used in this version of OpenToken is based on the SHA-1 hashing algorithm specified in [\[SHA\] \(National Institute of Standards and Technology, "Secure Hash Standard," August 2002.\)](#). See [Section 7 \(Security Considerations\)](#) for further information about the security characteristics of this algorithm.

*The Key Info field provides a place to store meta-data describing the key used to encrypt the payload. For example, it can contain a cryptographic hash of the key, or some other identifier, so that the token recipient can select the appropriate key for decryption. This field is OPTIONAL and MAY have a length of 0 (Key Info length) to indicate that no meta-data is available for this token.

Given the limited scope of applicability and the desire for a lightweight exchange format, OpenToken uses a binary format rather than a more generic data-description language such as [\[ASN.1\] \(CCITT,](#)

[“Recommendation X.208: Specification of Abstract Syntax Notation One \(ASN.1\),” 1988.](#)) or [\[XML\] \(Paoli, J., Maler, E., Sperberg-McQueen, C., Yergeau, F., and T. Bray, “Extensible Markup Language \(XML\) 1.0 \(Fourth Edition\),” August 2006.\)](#).

3. Processing Rules

[TOC](#)

3.1. Encoding

[TOC](#)

Generating an OTK involves the following steps:

1. Generate the payload.
2. Select a cipher suite and generate a corresponding IV.
3. Initialize an [\[HMAC\] \(National Institute of Standards and Technology, “The Keyed-Hash Message Authentication Code \(HMAC\),” March 2002.\)](#) using the SHA-1 algorithm specified in [\[SHA\] \(National Institute of Standards and Technology, “Secure Hash Standard,” August 2002.\)](#) and the following data (order is significant):
 1. OTK version
 2. Cipher suite value
 3. IV value (if present)
 4. Key Info value (if present)
 5. Payload length (2 bytes, network order)
4. Update the SHA-1 HMAC (from the previous step) using the clear-text payload.
5. Compress the payload using the DEFLATE specification in accordance with [\[RFC1950\] \(Deutsch, L. and J-L. Gailly, “ZLIB Compressed Data Format Specification version 3.3,” May 1996.\)](#) and [\[RFC1951\] \(Deutsch, P., “DEFLATE Compressed Data Format Specification version 1.3,” May 1996.\)](#).
6. Encrypt the compressed payload using the selected cipher suite.
7. Construct the binary structure representing the OTK; place the MAC, IV, key info and cipher-text within the structure.
8. Base 64 encode the entire binary structure, following the rules in Section 4 of [\[RFC4648\] \(Josefsson, S., “The Base16, Base32,](#)

[and Base64 Data Encodings," October 2006.](#)) and ensuring that the padding bits are set to zero.

9. Replace all Base 64 padding characters "=" with "*" (RFC 4648 does not account for the problems that Base64 padding causes when used as a cookie; this step corrects that issue).

3.2. Decoding

[TOC](#)

Processing an OTK involves the following steps:

1. Replace the "*" padding characters (see Encoding section, step 9) with standard Base 64 "=" characters.
2. Base 64 decode the OTK, following the rules in Section 4 of [\[RFC4648\] \(Josefsson, S., "The Base16, Base32, and Base64 Data Encodings," October 2006.\)](#) and ensuring that the padding bits are set to zero.
3. Validate the OTK header literal and version.
4. Extract the Key Info (if present) and select a key for decryption.
5. Decrypt the payload cipher-text using the selected cipher suite.
6. Decompress the decrypted payload, in accordance with [\[RFC1950\] \(Deutsch, L. and J-L. Gailly, "ZLIB Compressed Data Format Specification version 3.3," May 1996.\)](#) and [\[RFC1951\] \(Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3," May 1996.\)](#).
7. Initialize an [\[HMAC\] \(National Institute of Standards and Technology, "The Keyed-Hash Message Authentication Code \(HMAC\)," March 2002.\)](#) using the SHA-1 algorithm specified in [\[SHA\] \(National Institute of Standards and Technology, "Secure Hash Standard," August 2002.\)](#) and the following data (order is significant):
 1. OTK version
 2. Cipher suite value
 3. IV value (if present)
 4. Key Info value (if present)
 5. Payload length (2 bytes, network order)
8. Update the HMAC from the previous step with the clear-text payload (after decompressing).

9. Compare the HMAC from step 8 with the HMAC received in the OTK. If they do not match, halt processing.
10. Process the payload.

3.3. Standard Key-Value Pairs

TOC

The token payload contains key-value pairs, as described under [Section 5 \(Payload Format\)](#). These data pairs are used to describe the token presenter and can vary from application to application. In the interest of basic interoperability when exchanging an OTK, there is a small set of standardized data pairs.

Key Name	Value Format	Description
subject	string	Primary identifier for the token holder.
not-before	ISO 8601 datetime; yyyy-MM-ddTHH:mm:ssZ	Datetime when token was created; a token received before this datetime MUST be rejected as invalid.
not-on-or-after	ISO 8601 datetime; yyyy-MM-ddTHH:mm:ssZ	Datetime at which token will expire; a token received on or after this datetime MUST be rejected as invalid.
renew-until	ISO 8601 datetime; yyyy-MM-ddTHH:mm:ssZ	Datetime at which token MUST NOT automatically re-issued without further authentication; this can be viewed as a "session" lifetime.

The following rules apply:

- *All datetimes MUST be calculated relative to UTC (i.e., no timezone offsets).
- *The predefined key-value pairs MUST NOT be used for any purpose other than what is defined above and SHOULD be included with all issued OTKs.

4. Cipher Suites

[TOC](#)

A cipher suite groups a cryptographic cipher with a specific key size, cipher mode, and padding scheme. This grouping provides a convenient way of representing these inter-dependent options and also helps the implementor understand the exact cryptographic requirements for a given OTK.

ID	Cipher	Key Size	Mode	Padding	IV Length
0	Null	N/A	N/A	N/A	0
1	AES	256 bits	CBC	PKCS 5	16
2	AES	128 bits	CBC	PKCS 5	16
3	3DES	168 bits	CBC	PKCS 5	8

Note:

*The Null cipher is meant only for testing purposes. It MUST NOT be used in production environments, since the payload would be passed in the clear. When using the Null cipher, the SHA-1 MAC value MUST be replaced with a standard SHA-1 hash of the uncompressed payload.

*For cipher suites that do not require an IV, the IV length MAY be zero.

*For information regarding PKCS #5 padding, see [\[RFC2898\] \(Kaliski, B., "PKCS #5: Password-Based Cryptography Specification Version 2.0," September 2000.\)](#).

5. Payload Format

[TOC](#)

OTK uses a simple, line-based format for encoding the key-value pairs in the payload. The format is encoded with UTF-8 and thus is guaranteed to support the transport of multi-byte characters. The syntax for an OpenToken is defined as follows using the Augmented Backus-Naur Form as specified in [\[ABNF\] \(Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF," October 2005.\)](#).


```

line      = key "=" value CRLF
key       = [whitespace] identifier [whitespace]
whitespace = HTAB / SP
identifier = anychar
value     = [whitespace] data [whitespace] /
           [whitespace] quoted-data [whitespace]
data      = anychar / ""
           ; all characters
           ; can be empty to represent null values
quoted-data = "\" anychar "\" /
            '\ ' anychar '\ '
           ; double and single quotes must be
           ; escaped via preceding backslash
anychar    = %x20-%x3C / %x3E-%x7E / %xA0-D7FF / %xF900-FDCF
           / %xFDF0-FFEF / %x10000-1FFFFD / %x20000-2FFFFD
           / %x30000-3FFFFD / %x40000-4FFFFD / %x50000-5FFFFD
           / %x60000-6FFFFD / %x70000-7FFFFD / %x80000-8FFFFD
           / %x90000-9FFFFD / %xA0000-AFFFFD / %xB0000-BFFFFD
           / %xC0000-CFFFFD / %xD0000-DFFFFD / %xE1000-EFFFFD
           ; any Unicode character except "="

```

The following rules apply:

- *Key names might not be unique. OpenToken supports multiple values for a key name by simply adding another key-value pair.
- *Key names are case-sensitive. It is RECOMMENDED that all key names be lowercase and use hyphens to separate "words".
- *If the value for a key is or includes a Uniform Resource Identifier (URI), the characters "&" and "=" SHOULD be percent-encoded according to the rules specified in [\[URI\] \(Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier \(URI\): Generic Syntax," January 2005.\)](#).

6. Canonical Test Data

[TOC](#)

It is important to ensure interoperability across tokens generated by different implementations/languages. The following test cases can be used to test an implementation and ensure that it generates properly encoded tokens. These tests are not exhaustive, but do cover the basic cipher suites.

For each test case, the key that was used to generate the output is included in base64 encoding. The generated token is also base64 encoded, as specified above.

Each token has two name-value pairs present:

```
foo = bar
bar = baz
```

Note: In the following test data, the tokens are wrapped across two lines to fit and the "\" character is used to denote the point of line wrapping.

6.1. Test Case 1: AES-128

[TOC](#)

key:

```
a66C9MvM8eY4qJKyCXKW+w==
```

token:

```
UFRLAQK9THj0okLTUB663QrJFg5qA58IDhAb93ondvcx7sY6s44eszNqAAAgA5W8Dc\
4XZwtsZ4qV3_IDI-Zn2_yadHHIhkGqNV5J9kw*
```

6.2. Test Case 2: AES-256

[TOC](#)

key:

```
a66C9MvM8eY4qJKyCXKW+19PWDeuc3thDyuiumak+Dc=
```

token:

```
UFRLAQEujlLGEvmVKDKyvL1vaZ27qMYhTxDSAZwtaufqUff7GQXTjvWBAAAgJJGpTa\
7VOITap4uDZ_0kW_Kt4yYZ4BBQzw_NR2CNE-g*
```

[TOC](#)

6.3. Test Case 3: 3DES-168

key:

a66C9MvM8eY4qJKyC XKW+19PWDeuc3th

token:

UFRLAQNoCsuAwybXOSBpIc9ZvxQVx_3fhghqSjy-pNJpfgAAGG1GgJ79NhX431LRXA\
b9Mp5unR7XFWopzw**

7. Security Considerations

[TOC](#)

Recent research has shown that in select cases it is possible to compromise the hashes produced by the SHA-1 hashing algorithm (see [\[RFC4270\]](#) (Hoffman, P. and B. Schneier, "Attacks on Cryptographic Hashes in Internet Protocols," November 2005.)). However, the use to which SHA-1 is put in OpenToken, coupled with employment of a symmetric cipher key, will likely minimize the applicability of the attacks described in the literature. Furthermore, current estimates suggest that even with the recently-discovered attack, it would still take one year of computing by a government-sized entity to produce a collision.

Tokens SHOULD be exchanged over a secure transport (e.g., HTTP Over TLS as described in [\[RFC2818\]](#) (Rescorla, E., "HTTP Over TLS," May 2000.)) to minimize the possibility that a token can be intercepted by a man-in-the-middle.

8. References

[TOC](#)

8.1. Normative References

[TOC](#)

[ABNF]	Crocker, D. and P. Overell , "Augmented BNF for Syntax Specifications: ABNF," RFC 4234, October 2005 (TXT).
[HMAC]	National Institute of Standards and Technology, " The Keyed-Hash Message Authentication Code (HMAC) ," FIPS PUB 198, March 2002.
[RFC1950]	Deutsch, L. and J-L. Gailly, " ZLIB Compressed Data Format Specification version 3.3 ," RFC 1950, May 1996 (TXT , PS , PDF).
[RFC1951]	Deutsch, P. , " DEFLATE Compressed Data Format Specification version 1.3 ," RFC 1951, May 1996 (TXT , PS , PDF).

[RFC4648]	Josefsson, S., " The Base16, Base32, and Base64 Data Encodings ," RFC 4648, October 2006 (TXT).
[SHA]	National Institute of Standards and Technology, " Secure Hash Standard ," FIPS PUB 180-2, August 2002.
[URI]	Berners-Lee, T., Fielding, R., and L. Masinter, " Uniform Resource Identifier (URI): Generic Syntax ," STD 66, RFC 3986, January 2005 (TXT, HTML, XML).

8.2. Informative References

[TOC](#)

[ASN.1]	CCITT, "Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1)," 1988.
[RFC2119]	Bradner, S., " Key words for use in RFCs to Indicate Requirement Levels ," BCP 14, RFC 2119, March 1997.
[RFC2616]	Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, " Hypertext Transfer Protocol -- HTTP/1.1 ," RFC 2616, June 1999 (TXT, PS, PDF, HTML, XML).
[RFC2818]	Rescorla, E., " HTTP Over TLS ," RFC 2818, May 2000 (TXT).
[RFC2898]	Kaliski, B., " PKCS #5: Password-Based Cryptography Specification Version 2.0 ," RFC 2898, September 2000 (TXT).
[RFC2965]	Kristol, D. and L. Montulli, " HTTP State Management Mechanism ," RFC 2965, October 2000 (TXT, HTML, XML).
[RFC4270]	Hoffman, P. and B. Schneier, " Attacks on Cryptographic Hashes in Internet Protocols ," RFC 4270, November 2005 (TXT).
[SAML]	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.
[SASL]	Melnikov, A. and K. Zeilenga, " Simple Authentication and Security Layer (SASL) ," RFC 4422, June 2006 (TXT).
[XML]	Paoli, J., Maler, E., Sperberg-McQueen, C., Yergeau, F., and T. Bray, " Extensible Markup Language (XML) 1.0 (Fourth Edition) ," World Wide Web Consortium Recommendation REC-xml-20060816, August 2006 (HTML).

Authors' Addresses

[TOC](#)

	Dave Smith
	Individual
Email:	dizzyd@dizzyd.com
	Peter Motykowski
	Ping Identity
Email:	peter@motyka.org
	Yasir Faruqi
	Ping Identity
Email:	yfaruqi@pingidentity.com

	Peter Saint-Andre (editor)
	Individual
Email:	stpeter@jabber.org

Full Copyright Statement

[TOC](#)

Copyright © The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.