Abstract

This document specifies some cryptographic algorithms which will be used for the Mutual user authentication method for the Hyper-text Transport Protocol (HTTP).

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 November 19, 2012.

Copyright Notice

Copyright (c) 2012 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.
1. Introduction

This document specifies some algorithms for Mutual authentication protocol for Hyper-Text Transport Protocol (HTTP) [I-D.oiw-http-mutualauth].

1.1. 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 [RFC2119].

The terms "encouraged" and "advised" are used for suggestions that do not constitute "SHOULD"-level requirements. People MAY freely choose not to include the suggested items regarding [RFC2119], but complying with those suggestions would be a best practice; it will improve the security, interoperability, and/or operational performance.

The term "natural numbers" refers to the non-negative integers (including zero) throughout this document.

This document treats target (codomain) of hash functions to be natural numbers. The notation OCTETS(H(s)) gives a usual octet-string output of hash function H applied to string s.

2. Authentication Algorithms

This document specifies only one family of the authentication algorithm. The family consists of four authentication algorithms, which only differ in their underlying mathematical groups and security parameters. The algorithms do not add any additional parameters. The tokens for these algorithms are...
- iso-kam3-dl-2048-sha256: for the 2048-bit discrete-logarithm setting with the SHA-256 hash function.
- iso-kam3-dl-4096-sha512: for the 4096-bit discrete-logarithm setting with the SHA-512 hash function.
- iso-kam3-ec-p256-sha256: for the 256-bit prime-field elliptic-curve setting with the SHA-256 hash function.
- iso-kam3-ec-p521-sha512: for the 521-bit prime-field elliptic-curve setting with the SHA-512 hash function.

For discrete-logarithm settings, the underlying groups are the 2048-bit and 4096-bit MODP groups defined in [RFC3526], respectively. See Appendix A for the exact specifications of the groups and associated parameters. The hash functions H are SHA-256 for the 2048-bit group and SHA-512 for the 4096-bit group, respectively, defined in FIPS PUB 180-2 [FIPS.180-2.2002]. The representation of the parameters kc1, ks1, vkc, and vks is base64-fixed-number.

For the elliptic-curve settings, the underlying groups are the elliptic curves over the prime fields P-256 and P-521, respectively, specified in the appendix D.1.2 of FIPS PUB 186-3 [FIPS.186-3.2009] specification. The hash functions H, which are referenced by the core document, are SHA-256 for the P-256 curve and SHA-512 for the P-521 curve, respectively. The representation of the parameters kc1, ks1, vkc, and vks is hex-fixed-number.

Note: This algorithm is based on the Key Agreement Mechanism 3 (KAM3) defined in Section 6.3 of ISO/IEC 11770-4 [ISO.11770-4.2006] with a few modifications/improvements. However, implementers should use this document as the normative reference, because the algorithm has been changed in several minor details as well as major improvements.

2.1. Support Functions and Notations

The algorithm definitions use several support functions and notations defined below:

The integers in the specification are in decimal, or in hexadecimal when prefixed with "0x".

The two functions named octet() and OCTETS() are those defined in the core specification [I-D.oiwa-http-mutualauth].

Note: The definition of OCTETS() is different from the function GE2OS_x in the original ISO specification, which takes the shortest representation without preceding zeros.

All of the algorithms defined in this specification use the default functions defined in the core specification for computing the values pi, VK_c and VK_s.

2.2. Functions for Discrete-Logarithm Settings

In this section, an equation \((x / y \mod z)\) denotes a natural number \(w\) less than \(z\) that satisfies \((w \times y) \mod z = x \mod z\).

For the discrete-logarithm, we refer to some of the domain parameters by using the following symbols:

- \(q\): for "the prime" defining the MODP group.
- \(g\): for "the generator" associated with the group.
- \(r\): for the order of the subgroup generated by \(g\).
The function $J$ is defined as

$$J(\pi) = g^{\pi} \mod q.$$  

The value of $K_{c1}$ is derived as

$$K_{c1} = g^{S_{c1}} \mod q,$$

where $S_{c1}$ is a random integer within range $[1, r-1]$ and $r$ is the size of the subgroup generated by $g$. In addition, $S_{c1}$ MUST be larger than $\log(q)/\log(g)$ (so that $g^{S_{c1}} > q$).

The value of $K_{c1}$ SHALL satisfy $1 < K_{c1} < q-1$. The server MUST check this condition upon reception.

Let an intermediate value $t_1$ be

$$t_1 = H(\text{octet}(1) \mid \text{OCTETS}(K_{c1})),$$

the value of $K_{s1}$ is derived from $J(\pi)$ and $K_{c1}$ as:

$$K_{s1} = (J(\pi) \ast K_{c1}^{t_1})^{S_{s1}} \mod q$$

where $S_{s1}$ is a random number within range $[1, r-1]$. The value of $K_{s1}$ MUST satisfy $1 < K_{s1} < q-1$. If this condition is not held, the server MUST retry using another value for $S_{s1}$. The client MUST check this condition upon reception.

Let an intermediate value $t_2$ be

$$t_2 = H(\text{octet}(2) \mid \text{OCTETS}(K_{c1}) \mid \text{OCTETS}(K_{s1})),$$

the value $z$ on the client side is derived by the following equation:

$$z = K_{s1}^{((S_{c1} + t_2) / (S_{c1} \ast t_1 + \pi)) \mod r} \mod q.$$  

The value $z$ on the server side is derived by the following equation:

$$z = (K_{c1} \ast g^{t_2})^{S_{s1}} \mod q.$$  

### 2.3. Functions for Elliptic-Curve Settings

For the elliptic-curve setting, we refer to some of the domain parameters by the following symbols:

- $q$: for the prime used to define the group.
- $G$: for the defined point called the generator.
- $r$: for the order of the subgroup generated by $G$.

The function $P(p)$ converts a curve point $p$ into an integer representing point $p$, by computing $x \ast 2 + (y \mod 2)$, where $(x, y)$ are the coordinates of point $p$. $P'(z)$ is the inverse of function $P$, that is, it converts an integer $z$ to a point $p$ that satisfies $P(p) = z$. If such $p$ exists, it is uniquely defined. Otherwise, $z$ does not represent a valid curve point. The operator $+$ indicates the elliptic-curve group operation, and the operation $[x] \ast p$ denotes an integer-multiplication of point $p$: it calculates $p + p + ...$
(x times) \( \ldots + p \). See the literatures on elliptic-curve cryptography for the exact algorithms used for those functions (e.g. Section 3 of [RFC6090], which uses different notations, though.) \( 0_E \) represents the infinity point. The equation \( (x / y \mod z) \) denotes a natural number \( w \) less than \( z \) that satisfies \( (w * y) \mod z = x \mod z \).

The function \( J \) is defined as

\[
J(\pi) = [\pi] * G.
\]

The value of \( K_{c1} \) is derived as

\[
K_{c1} = P(K_{c1}'), \text{ where } K_{c1}' = [S_{c1}] * G,
\]

where \( S_{c1} \) is a random number within range \([1, r-1]\). The value of \( K_{c1} \) \textbf{MUST} represent a valid curve point, and \( K_{c1}' \) \textbf{SHALL NOT} be \( 0_E \). The server \textbf{MUST} check this condition upon reception.

Let an intermediate integer \( t_1 \) be

\[
t_1 = H(\text{octet}(1) | \text{OCTETS}(K_{c1})),
\]

the value of \( K_{s1} \) is derived from \( J(\pi) \) and \( K_{c1}' = P'(K_{c1}) \) as:

\[
K_{s1} = P([S_{s1}] * (J(\pi) + [t_1] * K_{c1}')), 
\]

where \( S_{s1} \) is a random number within range \([1, r-1]\). The value of \( K_{s1} \) \textbf{MUST} represent a valid curve point and satisfy \( 4 * P'(K_{s1}) \not< 0_E \). If this condition is not satisfied, the server \textbf{MUST} retry using another value for \( S_{s1} \). The client \textbf{MUST} check this condition upon reception.

Let an intermediate integer \( t_2 \) be

\[
t_2 = H(\text{octet}(2) | \text{OCTETS}(K_{c1}) | \text{OCTETS}(K_{s1})) ,
\]

the value \( z \) on the client side is derived by the following equation:

\[
z = P([S_{c1} + t_2] / (S_{c1} * t_1 + \pi) \mod r] * P'(K_{s1})).
\]

The value \( z \) on the server side is derived by the following equation:

\[
z = P([S_{s1}] * (P'(K_{c1}) + [t_2] * G)).
\]

### 3. IANA Considerations

Four tokens \texttt{iso-kam3-dl-2048-sha256}, \texttt{iso-kam3-dl-4096-sha512}, \texttt{iso-kam3-ec-p256-sha256} and \texttt{iso-kam3-ec-p521-sha512} shall be allocated and registered according to the provision of the core documentation when this document is promoted to an RFC.

Note: More formal declarations will be added in the future drafts to meet the RFC 5226 requirements.
4. Security Considerations

Refer the corresponding section of the core specification for algorithm-independent, generic considerations.

- All random numbers used in these algorithms MUST be at least cryptographically computationally secure against forward and backward guessing attacks.
- Computation times of all numerical operations on discrete-logarithm group elements and elliptic-curve points MUST be normalized and made independent of the exact values, to prevent timing-based side-channel attacks.

5. Notice on intellectual properties

The National Institute of Advanced Industrial Science and Technology (AIST) and Yahoo! Japan, Inc. has jointly submitted a patent application on the protocol proposed in this documentation to the Patent Office of Japan. The patent is intended to be open to any implementors of this protocol and its variants under non-exclusive royalty-free manner. For the details of the patent application and its status, please contact the author of this document.

The elliptic-curve based authentication algorithms might involve several existing third-party patents. The authors of the document take no position regarding the validity or scope of such patents, and other patents as well.

6. References

6.1. Normative References


6.2. Informative References

Appendix A. (Informative) Group Parameters for Discrete-Logarithm Based Algorithms

The MODP group used for the iso-kam3-dl-2048-sha256 algorithm is defined by the following parameters.

The prime is:

\[
q = 0xFF \text{FFFFF} \text{FFFFF} \text{FFFFF} \text{FFFFF} \ C90\text{DA} \text{A}2 \text{2168C2} \text{34} \text{C}4 \text{C}6 \text{628B} \text{80DC1CD}1 \\
29024E08 \text{8A67CC74} \text{020B8EA6} \text{3B139B22} \text{514A0879} \text{8E3404DD}
\]

The generator is:

\[
g = 2.
\]

The size of the subgroup generated by g is:

\[
r = (q - 1) / 2 = \\
0xFF \text{FFFFF} \text{FFFFF} \text{E487ED51} \text{10B4611A} \text{62633145} \text{C06E0E68} \\
94B13720 \text{453F5E6A} \text{0105D8F3} \text{1D89CD91} \text{28A5043C} \text{C71A026E}
\]

The MODP group used for the iso-kam3-dl-4096-sha512 algorithm is defined by the following parameters.

The prime is:

\[
q = 0xFF \text{FFFFF} \text{FFFFF} \text{C90FDAA}2 \text{2168C2} \text{34} \text{C4C6} \text{628B} \text{80DC1CD}1 \\
29024E08 \text{8A67CC74} \text{020B8EA6} \text{3B139B22} \text{514A0879} \text{8E3404DD}
\]

-7-
The generator is:

\[ g = 2. \]

The size of the subgroup generated by \( g \) is:

\[ r = \frac{(q - 1)}{2} = 0x7FFFFFFF FFFFFFFE 487249B0 \]

Appendix B. (Informative) Derived Numerical Values

This section provides several numerical values for implementing this protocol, derived from the above specifications. The values shown in this section are for informative purposes only.

<table>
<thead>
<tr>
<th>dl-2048</th>
<th>dl-4096</th>
<th>ec-p256</th>
<th>ec-p521</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of ( K_{c1} ) etc.</td>
<td>2048</td>
<td>4096</td>
<td>257</td>
</tr>
<tr>
<td>Size of ( H(...) )</td>
<td>256</td>
<td>512</td>
<td>256</td>
</tr>
<tr>
<td>length of OCTETS(( K_{c1} )) etc.</td>
<td>256</td>
<td>512</td>
<td>33</td>
</tr>
<tr>
<td>length of ( k_{c1}, k_{s1} ) param. values.</td>
<td>344 *</td>
<td>684 *</td>
<td>66</td>
</tr>
<tr>
<td>length of ( v_{kc}, v_{ks} ) param. values.</td>
<td>44 *</td>
<td>88 *</td>
<td>64</td>
</tr>
</tbody>
</table>
Appendix C. (Informative) Draft Change Log

C.1. Changes in revision 02

- Implementation hints in appendix changed (number of characters for base64-fixed-number does not contain double-quotes).

C.2. Changes in revision 01

- Parameter names renamed.
- Some expressions clarified without changing the value.

C.3. Changes in revision 00

The document is separated from the revision 08 of the core documentation.

Authors’ Addresses

Yutaka Oiwa
National Institute of Advanced Industrial Science and Technology
Research Institute for Secure Systems
Tsukuba Central 2
1-1-1 Umezono
Tsukuba-shi, Ibaraki
JP
Email: mutual-auth-contact-ml@aist.go.jp

Hajime Watanabe
National Institute of Advanced Industrial Science and Technology

Hiromitsu Takagi
National Institute of Advanced Industrial Science and Technology

Boku Kihara
Lepidum Co. Ltd.
#602, Village Sasazuka 3
1-30-3 Sasazuka
Shibuya-ku, Tokyo
JP

Tatsuya Hayashi
Lepidum Co. Ltd.

Yuichi Ioku