Mutual Authentication Protocol for HTTP:
KAM3-based Cryptographic Algorithms
draft-ietf-httpauth-mutual-algo-02

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 August 23, 2015.

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.
1. Introduction

This document specifies some algorithms for Mutual authentication protocol for Hyper-Text Transport Protocol (HTTP) [I-D.ietf-httpauth-mutual]. The algorithms are based on so-called "Augmented Password-based Authenticated Key Exchange" (Augmented PAKE) techniques. In particular, it uses one of three key exchange algorithm defined in the ISO 11770-4: "Key management - Mechanisms based on weak secrets" [ISO.11770-4.2006] as a basis.

In very brief summary, the Mutual authentication protocol exchanges four values, $K_{c1}$, $K_{s1}$, $VK_e$ and $VK_s$, to perform authenticated key exchanges, using the password-derived secret $pi$ and its "augmented version" $J(pi)$. This document defines the set of functions $K_{c1}$, $K_{s1}$, and $J$ for a specific algorithm family.

Please note that, from the view of cryptographic literature, the original functionality of Augmented PAKE is separated into the functions $K_{c1}$ and $K_{s1}$ defined in this draft, and the functions $VK_e$ and $VK_s$ defined in Section 11 of I-D.ietf-httpauth-mutual] as "default functions". For the purpose of security analysis, please also refer to these functions.
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 term "natural numbers" refers to the non-negative integers (including zero) throughout this document.

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

2. Cryptographic Overview (Non-normative)

The cryptographic primitive used in this algorithm specification is based on a variant of augmented PAKE proposed by T. Kwon, called APKAS-AMP, originally submitted to IEEE P1363.2. The general flow of the successful exchange is shown below, for informative purposes only. The DL-based notations are used, and all group operations (mod q and mod r) are omitted.

Note that the only messages corresponding to the earlier two exchanges are defined in this specification. Those for latter two exchanges are defined in the main specification [I-D.ietf-httpauth-mutual].

3. 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 hash iteration count nIterPi is 16384. 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 186-4 [FIPS.186-4.2013] 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. Cofactors of these curves are 1. The hash iteration count nIterPi is 16384. The representation of the parameters kc1, ks1, vkc, and vks is hex-fixed-number.

[[Editorial Note: remove before submission] We should take a care on recent hot discussion topic on the choice of elliptic curves for cryptography.]]

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.

### 3.1. 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 functions named octet(), OCTETS(), and INT() are those defined in the core specification [I-D.ietf-httpauth-mutual].

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 (defined in Section 11 of [I-D.ietf-httpauth-mutual]) for computing the values pi, VK_c and VK_s.

### 3.2. 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 * 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

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

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

\[ K_{s1} = (J(pi) * K_{c1}^{(t_1)})^{S_{s1}} \mod q \]

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 reject the exchange. The client MUST check this condition upon reception.

Let an intermediate value \( t_2 \) be

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

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

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

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

\[ z = (K_{c1} * g^{(t_2)})^{S_{s1}} \mod q. \]

(Note: the original ISO specification contained a message pair containing verification of value \( z \) along with the "transcript" of the protocol exchange. The functionality of this kind is contained in the functions VK\(_c\) and VK\(_s\).)
3.3. 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.
- h: for the cofactor of the group.
- 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 \( x \times 2 + (y \mod 2) \), 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 \([x] \times p\) denotes an integer-multiplication of point \( p \): it calculates \( p + p + \ldots \) (\( x \) times) \( + p \). 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 \((w \times y) \mod z = x \mod z \).

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 \( K_{c1} \) MUST represent a valid curve point, and \([h] \times K_{c1}' \) SHALL NOT be \( 0_E \). The server MUST check this condition upon reception.

Let an intermediate integer \( t_1 \) be

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

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

\[
K_{s1} = P([S_{s1}] * (J(pi) + [t_1] \times K_{c1}')), \text{ where } S_{s1} \text{ is a random number within range } [1, r-1].
\]

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

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

4. IANA Considerations

Four tokens iso-kam3-dl-2048-sha256, iso-kam3-dl-4096-sha512, iso-kam3-ec-p256-sha256 and 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.

5. Security Considerations

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

5.1. General Implementation Considerations

- During the exchange, the value \( VK_{s} \), defined in [I-D.ietf-httpauth-mutual], MUST only be sent when the server has received a correct (expected) value of \( VK_{c} \). This is a requirement from underlying cryptography stated in [ISO.11770-4:2006].
- 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.

5.2. Cryptographic Assumptions and Considerations

The notices on this subsection is mostly for those who analyze the security of this algorithm, and those who might want to make a derived work of this algorithm specification.

- Handling of invalid \( K_{s1} \) value in the exchange (now: to reject the exchange) has been changed from original ISO specification (original: to retry with another random \( S_{s1} \) value). This is due to an observation that this condition is less likely from the random error caused by unlucky choice of \( S_{s1} \), but more likely from the systematic failure from invalid \( J(pi) \) value, even implying possible denial-of-service attacks.
- The usual construction of authenticated key exchange algorithms are build from a key-exchange period and a key verification period, and the latter usually involving some kind of exchange transaction to be verified, to avoid security risks or vulnerabilities caused from mixing of values from two or more key exchanges. In the design of the algorithms in this document, such a functionality is defined in generalized manner in the core specification [I-D.ietf-httpauth-mutual] (see definitions of \( VK_{c} \) and \( VK_{s} \)). If any attempts to reuse the algorithm defined above with any other protocols exist, care MUST be taken on that aspect.
- The domain parameters chosen and specified in this draft has a few assumptions. In the DL setting, \( q \) has to be safe prime \(((q - 1) / 2) \) must also be prime), and \( r \) should be the largest possible value \(((q - 1) / 2) \). In the EC setting, \( r \) has to be prime. Defining a variation of this
algorithm using a different domain parameter SHOULD care about these conditions.

6. 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 implementers 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.

7. References

7.1. Normative References


7.2. 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:

\[ q = \text{0x}FFFFFFFFFF \text{ FFFFFFFF C90FDAA2 2168C234 C46622B8 0DC1CD1} \]

29024E08 8A67CC74 020B0EA6 3B139B22 51A40879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 62E7ECE6 F4C42E9 A637ED6B 08FF5CB6 F406B7ED
EE386BFB 5A999FA5 AE9F2411 74CB1FE6 49286651 ECE45B3D
C2007CB8 A163BF05 98ADA4836 1C55D39A 96163FAB FD24CF5F
83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F174C08 CA18217C 32905E46 2E36CE3B
E39E772C 180E8603 9B27B3A2 EC07A28F B5C55DF0 6F4C52C9
DE2BCF6 95581718 3995497C EA956AE5 15D22618 98FA0510
15728E5A 8AACAA68 FFFFFFFF FFFFFFFF.

The generator is:

\[ g = 2. \]

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

\[ r = (q - 1) / 2 = \text{0x}7FFFFFFF FFFFFFFF E487ED51 10B4611A 62633145 C06DE0E8 \]

94812704 4533E33A 0105DF53 1D89CD91 28A5043C C71A026E
F7C8C9D9 E69D218D 98158536 F92F8A1B A7F09AB6 B6A8E122
F2422AB8 312F6F3 7A262174 D31B6F85 85FFAE5B 7A035BF6
F71C35FD AD4C4FD2 D74F9208 BE2588F3 24943328 F6722D9E
E1003E5C 50BD1F82 CC6D241B 0E2AE9CD 34881FD4 7E9267AF
C12B2A69 E51D6CB 0E3179AB 1042A95D C6F9A483 B84B4B36
B3861AA7 255E4C02 78BAA3604 650C10BE 19482F23 171B671D
F1CFC896 C074301 D93C1D1 7603D147 DAE2AEF8 37A62964
EF15E5FB 4AC00B8C 1CCAA4BE 754AB572 8AE9130C 4C7D0288
0AB9472D 45565534 7FFFFFFF FFFFFFFF.

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

The prime is:

\[ q = \text{0x}FFFFFFFFFF \text{ FFFFFFFF C90FDAA2 2168C234 C46622B8 0DC1CD1} \]

29024E08 8A67CC74 020B0EA6 3B139B22 51A40879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 62E7ECE6 F4C42E9 A637ED6B 08FF5CB6 F406B7ED
EE386BFB 5A999FA5 AE9F2411 74CB1FE6 49286651 ECE45B3D
C2007CB8 A163BF05 98ADA4836 1C55D39A 96163FAB FD24CF5F
83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F174C08 CA18217C 32905E46 2E36CE3B
E39E772C 180E8603 9B27B3A2 EC07A28F B5C55DF0 6F4C52C9
DE2BCF6 95581718 3995497C EA956AE5 15D22618 98FA0510
15728E5A 8AACAA68 FFFFFFFF FFFFFFFF.
The generator is:

\[ g = 2. \]

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

\[ r = (q - 1) / 2 = \]

0x7FFFF000 FFFFFFFF E487ED51 10B4611A 62633145 C06E0E68 049812704 4533E63A 0105DF53 1D89CD91 28A5043C C71A026E F7CA8CD9 E69D21BD 98158536 F92F8A1B A7F09AB6 B6A8E122 F242DABB 312F3F63 7A262174 D31BF6B5 85FFA5E5B 7A035BF6 F71C35FD AD44CFD2 D74F9208 BE258F3 C94533E63A 0105DF53 1D89CD91 28A5043C C71A026E E1003E5C 50B1DF82 CC6D241B 0E2AE9CD 348B1FD4 7E9267AF C12BA9E91 E51D6CB 0E3179AB 1042A95D CF6A9483 B84B4B36 B3861AA7 255E4C02 78BA3604 690C10BE 19482F23 171B671D F1CF3B96 0C074301 CD93C1D1 7603D147 DAE2AEC8 37A62964 EF15E5FB 4AAC0B8C 1CCAA9BE 754AB572 8AE9130C 4C7DD028 0AB9472D 45562616 D699BB86 82283D19 D24A905D EF85ED32 767DC282 2C6DF785 457538AB AE83063E 9CB87C2 D370F263 D5FAD746 6D8499EB 8F464A70 2512B0CE E771E913 0D697735 F897FD03 6CC50432 6C3B0139 9F643532 290F958C 0BBDD906 5DF088AB BD30AE6B 3B84C460 5D6CA371 047127D0 3A72598 A1EDAFDE 707E8847 25C16890 54908400 8D391E09 53C3F36B C438CD08 5EDD2D93 4CE1938C 357A711E 0D4A341A 5B0A85ED 12CF14E5 156A2674 6DDDE16D 826F477C 97477E0A 0DF6553 143E2CA3 A735E02E CCD94B27 D0488D1D 119D00C3 28ADF3F6 8FB094B8 67716BD7 DC0DEEBB 10B8240E 68034893 EAD82D54 C9DA8754C 46C7EEE0 C37FDBEE 48536047 A6F1A1AE 9A0318CC FFFFFFFF FFFFFFFF.

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></th>
<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>
<td>522 (bits)</td>
</tr>
<tr>
<td>hSize, Size of ( H(...) )</td>
<td>256</td>
<td>512</td>
<td>256</td>
<td>512 (bits)</td>
</tr>
<tr>
<td>length of ( OCTETS(K_{c1}) ) etc.</td>
<td>256</td>
<td>512</td>
<td>33</td>
<td>66 (octets)</td>
</tr>
<tr>
<td>length of ( kc1, ks1 ) param. values.</td>
<td>344 *</td>
<td>684 *</td>
<td>66</td>
<td>132 (octets)</td>
</tr>
<tr>
<td>length of ( vkc, vks ) param. values.</td>
<td>44 *</td>
<td>88 *</td>
<td>64</td>
<td>128 (octets)</td>
</tr>
<tr>
<td>minimum allowed ( S_{c1} )</td>
<td>2048</td>
<td>4096</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(The numbers marked with an * do not include any enclosing quotation marks.)

Appendix C. (Informative) Draft Change Log
C.1. Changes in HTTPAUTH-WG revision 02
- No technical changes: references updated.

C.2. Changes in HTTPAUTH-WG revision 01
- Changed behavior on failed generation of $K_{s1}$.
- Security considerations updated.

C.3. Changes in HTTPAUTH-WG revision 00
- Added a note on the choice of elliptic curves.

C.4. Changes in HTTPAUTH revision 02
- Added nIterPi parameter to adjust to the changes to the core draft.
- Added a note on the verification of exchange transaction.

C.5. Changes in HTTPAUTH revision 01
- Notation change: integer output of hash function will be notated as INT(H(*)), changed from $H(*)$.

C.6. Changes in revision 02
- Implementation hints in appendix changed (number of characters for base64-fixed-number does not contain double-quotes).

C.7. Changes in revision 01
- Parameter names renamed.
- Some expressions clarified without changing the value.

C.8. 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
3-11-46 Nakouji
Amagasaki, Hyogo
JP
Email: mutual-auth-contact-ml@aist.go.jp

Hajime Watanabe
Hiromitsu Takagi
National Institute of Advanced Industrial Science and Technology
Research Institute for Secure Systems
Tsukuba Central 2
1-1-1 Umezono
Tsukuba-shi, Ibaraki
JP

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

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

Yuichi Ioku
Individual