

Internet Draft

David M'Raïhi
VeriSign
Johan Rydell
Portwise
David Naccache
ENS
Salah Machani
Diversinet

Category:
Informational

Document:
draft-mraihi-mutual-oath-hotp-variants-00.txt

Expires:
June 2006

December 2005

Mutual OATH: HOTP Extensions for mutual authentication

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/1id-abstracts.html>
The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

Abstract

This document describes Mutual OATH, a mechanism for mutual authentication based on the HOTP algorithm [HOTP] introduced recently by OATH (initiative for Open AuThentication) [OATH] and submitted as an individual draft to the IETF last year. The security and strength of Mutual OATH depends on the properties of the underlying building block HOTP, which is a construction based on HMAC [RFC2104] using SHA-1 as the hash function.

Table of Contents

1. Introduction..... 2
Page 1

- 2. Requirements Terminology..... 3
- 3. Algorithm Requirements..... 3
- 4. Mutual OATH Definition..... 4
- 4.1 HOTP Algorithm..... 4
- 4.2 Algorithm Identifier..... 5
- 4.3 Mutual OATH Authentication..... 6
- 4.4 Dual-key Mode..... 6
- 4.5 Chained Mode..... 7
- 4.6 PIN/Password "salted" HOTP Response..... 7
- 4.7 Signature - Using HOTP with challenge and fixed value..... 8
- 5. Security Considerations..... 8
- 6. IANA Considerations..... 9
- 7. Conclusion..... 9
- 8. Acknowledgements..... 9
- 9. References..... 9
- 10.1 Normative..... 9
- 10.2 Informative..... 9
- 10. Authors' Addresses..... 10
- 12. Full Copyright Statement..... 11
- 13. Intellectual Property..... 11

1. Introduction

The HOTP algorithm is counter based and intended for synchronous authentication mode systems. It has been implemented under various form factors such as USB tokens, software client application, validation servers, applets for smart-cards, etc.

OATH has identified several user cases and scenarios that require an asynchronous variant to accommodate users who do not want to maintain a synchronized authentication system. The commonly accepted method for this is to use a challenge-response scheme.

This draft introduces the concept of randomized versions of HOTP, where the counter field is replaced or extended by a function of several parameters such as a random question and a personal identification code or password.

Challenge response mode of authentication is widely adopted in the industry. Several vendors already offer software applications and hardware devices implementing challenge-response - but each of those uses vendor-specific proprietary algorithms. For the benefits of users we need a standardized challenge-response algorithm to

allow multi-sourcing of token purchases and validation systems to facilitate the democratization of strong authentication.

2. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14.

3. Algorithm Requirements

This section presents the main requirements that drove this

draft-mraihi-mutual-oath-hotp-variants-00.txt
algorithm design. A lot of emphasis was placed on flexibility and usability, under the constraints and specificity of the HOTP algorithm and hardware token capabilities.

R1 - The shared secret MUST be embedded in a tamper resistance device or securely implemented in a software application.

R2 - The secret SHOULD be stored in a hardware device for additional flexibility (mobility) and security.

R3 - The algorithm MUST be capable of supporting different modes of authentications.

R4 - The challenge value MUST be randomly generated for each use of the authentication protocol and SHALL NOT be re-used.

R5 - The algorithm MUST allow for the challenge to be presented to the user in numeric or alphanumeric form.

R6 - If the challenge is displayed to the user, the challenge length SHOULD be selectable; the specification should define the maximum length for a challenge.

R7 - There MUST be a fixed randomly generated secret (key) for each token/soft token that is shared between the token and the authentication server.

R8 - The challenge MAY be generated with integrity checking (e.g., parity bits). This will allow tokens with pin pads to perform simple error checking if the user enters the value into a token.

R9 - The algorithm MAY include a counter function.

R10 - The algorithm SHOULD be used in dual-key mode. The single key is mainly to support legacy tokens.

R11 - All the HOTP communications MUST take place over a secure channel e.g. SSL/TLS, IPsec connections.

4. Mutual OATH Definition

In this section, we introduce the HOTP algorithm variants for mutual authentication.

Words used in this chapter:

K - Key, a shared secret known to all parties

C - Counter, a predefined sequence of values that can be calculated by all parties

P - Hashed version of PIN/password that is known to all parties

Q - Challenge, a predefined value that may change and is known to all parties during execution of the algorithm

T - Fixed value that is used in a transaction

AI - Algorithm identifier, describes in detail in section 4.2

4.1 HOTP Algorithm

The HOTP algorithm is based on an increasing counter value and a static symmetric key known only to the prover and verifier parties. HOTP has been introduced as an internet draft and is currently in the RFC-editor queue for final review.

As a reminder:

$$\text{HOTP}(K, C) = \text{Truncate}(\text{HMAC-SHA-1}(K, C))$$

Where Truncate represents the function that converts an HMAC-SHA-1 value into an HOTP value of 6 to 8 digits.

The Key (K), the Counter (C) and Data values are hashed high-order byte first. The HOTP values generated by the HOTP generator are treated as big endian.

We refer the reader to [HOTP] for the full description and further details on the rationale and security analysis of HOTP.

We introduced in [HOTP] the notion of a Data field that would be used for generating the One-Time Password values. Using a Data field opens for more flexibility in the algorithm implementation, provided that the Data field is clearly specified.

A straightforward variant is to replace the counter value C by a random question Q to define a Response as a function of the challenge Q:

$$\text{Response} = \text{HOTP}(K, Q) = \text{Truncate}(\text{HMAC-SHA-1}(K, Q))$$

The present draft describes the different variants based on similar constructions.

4.2 Algorithm Identifier

It is important for both parties willing to interact to know the operations to be performed, namely which variant is to be used.

Let AI be a binary string, and the different bits indicate which combination of parameters is used to compute an HOTP response.

We could also specify the algorithm itself - by default, HOTP is based on HMAC-SHA-1 but recent attacks on hash functions and new standard works might result in recommendations to use another function for HMAC.

We define the algorithm identifier AI as follows:

$$\text{AI} = \text{Type} | \text{Pb} | \text{Qb} | \text{Cb} | \text{Tb} | \text{Mode}$$

Where:

Type is a 4-bit value to specify the algorithm type; at the moment, 4 types are defined:

- 0000: HMAC-SHA-1 without truncation
- 0001: HMAC-SHA-1 with Dynamic Truncation for 6-digit
- 0010: HMAC-SHA-1 with Dynamic Truncation for 7-digit

draft-mraihi-mutual-oath-hotp-variants-00.txt

0011: HMAC-SHA-1 with Dynamic Truncation for 8-digit

Pb - single bit that indicates whether or not a hash of a PIN or Password will be used in the computation;

Qb - single bit that indicates whether or not a random Q will be used in the computation;

Cb - single bit that indicates whether or not a counter C will be used in the computation;

Tb - single bit that indicates whether or not a fixed value T will be used in the computation.

Mode is a 4-bit value that defines the different modes of computation; at the moment, 4 types are defined:

0000: plain mode, single key

0001: plain mode, dual key

0010: chained mode, single key

0011: chained mode, dual key

With the chained mode as described in section 4.5

For instance, if the algorithm used is HOTP (K, C) with Dynamic Truncation to generate 6-digit HOTP values as described in [HOTP] as default mode then AI = 0001 0010 0000.

4.3 Mutual OATH Authentication

A straightforward variant is to replace the counter value C by a random question Q to define a randomized Response as a function of the challenge Q:

Response = HOTP (K, Q) = Truncate (HMAC-SHA-1(K, Q))

The challenge-response sequence is the following for parties A and B sharing knowledge of secret (key) K and willing to perform Mutual OATH authentication:

- 1- A sends a random question Q_A to B and AI = 0001 0100 0000
- 2- B computes Response_B = HOTP (K, Q_A) and sends it to A with his own random question Q_B
- 3- A checks Response_B and if correct, computes Response_A = HOTP (K, Q_B)
- 4- B checks Response_A and if correct, acknowledges party A

Both parties are authenticated.

4.4 Dual-key Mode

The key SHOULD be pre-divided into different usages. K1 is used for computing responses and K2 to check responses. The opposite is used for the other party. Depending on whether storage or computation is cheaper, we could either store a longer key $K = K1 \parallel K2$ or set $K1 = K$ and $K2 = K \text{ xor Constant}$ or $K2 = \text{SHA-1}(K)$ for instance.

The challenge-response sequence is the following for parties A and B sharing knowledge of secret (key) K. K is divided into K1 and K2.

- 1- A sends a random question Q_A to B and AI = 0001 0100 0001
- 2- B computes Response_B = HOTP (K2, Q_A) and sends it to A with his own random question Q_B
- 3- A checks Response_B and if correct, computes Response_A =

HOTP (K1, Q_B)

- 4- B checks Response_A and if correct, acknowledges party A

Both parties are authenticated. Separating into K1 and K2 protects against another party sending predefined questions "Q_A"'s to B for querying specific information.

We recommend this mode of computation for mutual OATH.

4.5 Chained Mode

Another interesting computation mode is to chain the computations of different responses. It might have valuable applications such as to verify that a certain sequence of operations has taken place.

The challenge-response sequence is the following for parties A and B sharing knowledge of secret (key) K and willing to perform Mutual OATH authentication in chained mode:

- 1- A sends a random question Q_A to B and AI = 0001 0100 0011
- 2- B computes Response_B = HOTP (K2, Q_A) and sends it to A with his own random question Q_B
- 3- A checks Response_B and if correct, computes Response_A = HOTP (K1, Q_B|Response_B)
- 4- B checks Response_A and if correct, acknowledges party A

Both parties are authenticated.

4.6 PIN/Password "salted" HOTP Response

In this case, the PIN/password value used to control the usage of the token and/or protect access to the key embedded in the hardware (or software) token can be injected in the Response computation, as well as the random question Q:

$$\text{Response} = \text{HOTP} (K, Q|P)$$

Where | stands for concatenation.

The challenge-response sequence is the following for parties A and B sharing knowledge of secret (key) K and willing to perform Mutual OATH authentication:

- 1- A sends a random question Q_A to B and AI = 000110000
- 2- B computes Response_B = HOTP (K2, Q_A|P) and sends it to A with his own random question Q_B
- 3- A checks Response_B and if correct, computes Response_A = HOTP (K1, Q_B|P)
- 4- B checks Response_A and if correct, acknowledges party A

Obviously, the string P can be empty if party A or B does not have a PIN or Password - e.g. a validation server computing a response to be authenticated by a user will probably not have the usage of a

PIN or password.

It opens for interesting combinations where the algorithm identifier AI could be used to specify computations both ways - e.g. the token could include the PIN in his computation and the

server, knowing the hash checks the HOTP response; in its calculation on the other hand, no PIN value would be included since the server would not have one.

4.7 Signature - Using HOTP with challenge and fixed value

The combination of a fixed value and random value enables to generate different signature values for different fixed values - the computation is randomized by the question Q. This is important when signing the same value more than once.

Signature = HOTP (K, Q|T) where | stands for concatenation.

The sequence for B generating a signature to be checked by A is the following, assuming the mode of computation is dual-key plain mode:

- 1- A sends a fixed value C, a random question Q to B and AI = 0001 0110 0001
- 2- B computes Response_B = HOTP (K2, Q|T) and sends it to A
- 3- A checks Response_B and if correct, acknowledges party B has validated the value T

In an hostile environment a third party can trick party B to reveal the correct response for a given value. Again, all exchanges should take place over a secure channel - using SSL/TLS, or similar encryption mechanism.

5. Security Considerations

The keys for HOTP can be of any length equal or longer than 20 bytes. Keys longer than 20 bytes are acceptable; they are first hashed using the supported hash function, e.g. SHA-1, to become usable. Nevertheless, the extra length would not significantly increase the cryptographic strength of Mutual OATH, provided the randomness of the original key material is sufficient.

Keys need to be chosen at random or using a cryptographically strong pseudo-random generator properly seeded with a random value. We RECOMMEND to follow the recommendations in [RFC1750] for all pseudo-random and random generations. The pseudo-random numbers used for generating the keys SHOULD successfully pass the randomness test specified in [CN].

The conclusion of the security analysis detailed in [HOTP] is that, for all practical purposes, the outputs of the dynamic truncation on distinct counter inputs are uniformly and independently distributed strings.

The analysis demonstrates that the best possible attack against the HOTP function is the brute force attack.

6. IANA Considerations

This document has no actions for IANA.

7. Conclusion

This draft introduced several variants of HOTP for randomized response and short signature-like computations.

The algorithm identifier AI provides for an easy integration and support of different HOTP flavors within an authentication and validation system.

Mutual OATH should enable cross-authentication both in connected and off-line modes, with the support of different response sizes and mode of operations.

8. Acknowledgements

We would like to thank Siddharth Bajaj, Philip Hoyer, Jon Martinsson and Stu Vaeth for their comments and suggestions to improve this draft document.

9. References

10.1 Normative

- [RFC2104] M. Bellare, R. Canetti and H. Krawczyk, "HMAC: Keyed-Hashing for Message Authentication", IETF Network Working Group, RFC 2104, February 1997.
- [RFC1750] D. Eastlake, 3rd., S. Crocker and J. Schiller, "Randomness Recommendations for Security", IETF Network Working Group, RFC 1750, December 2004.
- [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3668] S. Bradner, "Intellectual Property Rights in IETF Technology", BCP 79, RFC 3668, February 2004.

10.2 Informative

- [HOTP] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache and O. Ranen, HOTP: HOTP: An HMAC-based One Time Password Algorithm", Internet Draft, Informational.
<http://www.ietf.org/internet-drafts/draft-mraihi-oath-hmac-otp-04.txt>

- [BCK] M. Bellare, R. Canetti and H. Krawczyk, "Keyed Hash Functions and Message Authentication", Proceedings of Crypto'96, LNCS Vol. 1109, pp. 1-15.

draft-mraihi-mutual-oath-hotp-variants-00.txt
[OATH] Initiative for Open AuTHentication
<http://www.openauthentication.org>

[CN] J. S. Coron and D. Naccache, "An accurate evaluation of Maurer's universal test" by Jean-Sebastien Coron and David Naccache In Selected Areas in Cryptography (SAC '98), vol. 1556 of Lecture Notes in Computer Science, S. Tavares and H. Meijer, Eds., pp. 57-71, Springer-Verlag, 1999

10. Authors' Addresses

Primary point of contact (for sending comments and question):

David M' Raihi
VeriSign, Inc.
685 E. Middlefield Road Phone: 1-650-426-3832
Mountain View, CA 94043 USA Email: dmraihi@verisign.com

Other Authors' contact information:

Johan Rydel
Portwise, Inc.
624 Ellis Street, Suite 102 Phone: 1-650-472-3582
Mountain View, CA 94043 USA Email: johan.rydel@portwise.com

David Naccache
ENS, DI
45 rue d'Ulm Phone: +33 6 16 59 83 49
75005, Paris France Email: david.naccache@ens.fr

Salah Machani
Diversinet Corp.
2225 Sheppard Avenue East
Suite 1801
Toronto, Ontario M2J 5C2 Phone: 1-416-756-2324 Ext. 321
Canada Email: smachani@diversinet.com

OATH-HMAC-OTP Expires - April 2005
HOTP: An HMAC-based One Time Password Algorithm

[Page 10] ♀
December 2005

12. Full Copyright Statement

Copyright (C) The Internet Society (2005).

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 AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES,

draft-mraihi-mutual-oath-hotp-variants-00.txt
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.

13.

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.