Internet Draft   IPsec Working Group
June 2002   S. Frankel, NIST
Expiration Date: December 2002   S. Kelly, Black Storm Networks
Category: Experimental

The HMAC-SHA-256-128 Algorithm and Its Use With IPsec

<draft-ietf-ipsec-ciph-sha-256-01.txt>

Status of this Memo

This document is a submission to the IETF Internet Protocol Security (IPsec) Working Group. Comments are solicited and should be addressed to the working group mailing list (ipsec@lists.tislabs.com) or to the editors.

Distribution of this memo is unlimited.

Abstract

This document describes the use of the HMAC algorithm in conjunction with the SHA-256 algorithm as an experimental authentication mechanism within the context of the IPsec AH and ESP protocols. This algorithm is intended to provide data origin authentication and integrity protection. Given the current lack of practical experience with SHA-256, implementations based on this document will be experimental in nature, and implementation is not required in order to claim compliance with the IPsec proposed standards. The version of the HMAC-SHA-256 authenticator described in this document specifies truncation to 128 bits, and is therefore named HMAC-SHA-256-128.

Table of Contents

1. Specification of Requirements

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" that appear in this document are to be interpreted as described in [RFC-2119].

2. Introduction

This document specifies the use of SHA-256 [SHA2-1] combined with HMAC [HMAC] as an experimental keyed authentication mechanism within the context of the IPsec AH and ESP protocols. This algorithm is intended to provide data origin authentication and integrity protection. Given the current lack of practical experience with SHA-256, implementations based on this document will be experimental in nature, and implementation is not required in order to claim compliance with the IPsec proposed standards. Furthermore, HMAC-SHA-1-96 [HMAC-SHA] provides sufficient security at a lower computational cost. The version of the HMAC-SHA-256 authenticator described in this document specifies truncation to 128 bits, and is therefore named HMAC-SHA-256-128. For further information on ESP, refer to [ESP] and [ROADMAP]. For further information on AH, refer to [AH] and [ROADMAP].

The goal of HMAC-SHA-256-128 is to ensure that the packet is authentic and cannot be modified in transit. Data integrity and data origin authentication as provided by HMAC-SHA-256-128 are dependent upon the scope of the distribution of the secret key. If the key is known only by the source and destination, this algorithm will provide both data origin authentication and data integrity for packets sent between the two parties. In addition, only a party with the identical key can verify the MAC.

3. The HMAC-SHA-256-128 Algorithm

[SHA2-1] and [SHA2-2] describe the underlying SHA-256 algorithm, while [HMAC] describes the HMAC algorithm. The HMAC algorithm provides a framework for inserting various hashing algorithms such as SHA-256.

The following sections contain descriptions of the various characteristics and requirements of the HMAC-SHA-256-128 algorithm.

3.1 Keying Material

HMAC-SHA-256-128 is a secret key algorithm. While no fixed key length is specified in [HMAC], for use with either ESP or AH a fixed key length of 256-bits MUST be supported. Key lengths other than 256-bits MUST NOT be supported (i.e. only 256-bit keys are to be used by

HMAC-SHA-256-128). A key length of 256-bits was chosen based on the recommendations in [HMAC] (i.e. key lengths less than the authenticator length decrease security strength and keys longer than the authenticator length do not significantly increase security strength).

[HMAC] discusses requirements for key material, which includes a discussion on requirements for strong randomness. A strong pseudo-random function MUST be used to generate the required 256-bit key.

At the time of this writing there are no specified weak keys for use with HMAC. This does not mean to imply that weak keys do not exist.

[ARCH] describes the general mechanism for obtaining keying material when multiple keys are required for a single SA (e.g. when an ESP SA requires a key for confidentiality and a key for authentication).

In order to provide data origin authentication, the key distribution mechanism must ensure that unique keys are allocated and that they are distributed only to the parties participating in the communication.

[HMAC] makes the following recommendation with regard to rekeying: "Current attacks do not indicate a specific recommended frequency for key changes ... However, periodic key refreshment is a fundamental security practice that helps against potential weaknesses of the function and keys, and limits the damage of an exposed key." Rekeying also reduces the information available to a cryptanalyst.

3.2 Padding

HMAC-SHA-256-128 operates on 512-bit blocks of data. Padding requirements are specified in [SHA2-1] and are part of the SHA-256 algorithm. If you build SHA-256 according to [SHA2-1] you do not need to add any additional padding as far as HMAC-SHA-256-128 is concerned. With regard to "implicit packet padding" as defined in [AH], no implicit packet padding is required.

3.3 Truncation

HMAC-SHA-256-128 produces a 256-bit authenticator value. This 256-bit value can be truncated as described in [HMAC]. For use with either ESP or AH, a truncated value using the first 128 bits MUST be supported. Upon sending, the truncated value is stored within the authenticator field. Upon receipt, the entire 256-bit value is computed and the first 128 bits are compared to the value stored in the authenticator field. No other authenticator value lengths are supported by HMAC-SHA-256-128.

The length of 128 bits was selected because it meets the security requirements described in [HMAC]. [HMAC] discusses the potential additional security which is provided by the truncation of the resulting MAC. Specifications which include HMAC are strongly encouraged to perform this MAC truncation.

3.4 Interaction with the ESP Cipher Mechanism

As of this writing, there are no known issues which preclude the use of the HMAC-SHA-256-128 with any specific cipher algorithm.

3.5 Performance

[HASH] states that "(HMAC) performance is essentially that of the underlying hash function". As of this writing no detailed performance analysis has been done of SHA-256, HMAC or HMAC combined with SHA-256.

[HMAC] outlines an implementation modification which can improve per-packet performance without affecting interoperability.

3.6 Test Vectors

The following test cases for HMAC-SHA-256 and HMAC-SHA-256-128 include the key, the data, and the resulting HMAC. The values of keys and data are either hexadecimal numbers (prefixed by "0x") or ASCII character strings (surrounded by double quotes). If a value is an ASCII character string, then the HMAC computation for the corresponding test case DOES NOT include the trailing null character ('\0') of the string. The computed HMAC values are all hexadecimal numbers.

These test cases were verified using 3 independent implementations: an HMAC wrapper on top of Aaron Gifford's SHA256 implementation (www.aarongifford.com/computers/sha.html), the BeeCrypt crypto library (www.virtualunlimited.com/products/beecrypt) and the Nettle cryptographic library (www.lysator.liu.se/~nisse/nettle). Partial blocks were padded as specified in [SHA2-1].

Test cases 1 and 2 were taken from the SHA-2 FIPS [SHA2-1] and test cases 4-10 were borrowed from [HMAC-TEST] with some key sizes adjusted for HMAC-SHA-256. These test cases illustrate HMAC-SHA-256 with various combinations of input and keysize. All test cases include the computed HMAC-SHA-256; only those with a keysize of 32 bytes (256 bits) also include the truncated HMAC-SHA-256-128.

```
Test Case #1: HMAC-SHA-256 with 3-byte input and 32-byte key
Key_len         : 32
Key             : 0x0102030405060708090a0b0c0d0e0f10
                    1112131415161718191a1b1c1d1e1f20
Data_len        : 3
Data            : "abc"
HMAC-SHA-256    : 0xa21b1f5d4cf4f73a4dd939750f7a066a
                    7f98cc131cb16a6692759021cfab8181
HMAC-SHA-256-128: 0xa21b1f5d4cf4f73a4dd939750f7a066a

Test Case #2: HMAC-SHA-256 with 56-byte input and 32-byte key
Key_len         : 32
Key             : 0x0102030405060708090a0b0c0d0e0f10
                    1112131415161718191a1b1c1d1e1f20
Data_len        : 56
```

```
   Data            : "abcdbcdecdefdefgefghfghighijhijk
                      ijkljklmklmnlmnomnopnopq"
   HMAC-SHA-256    : 0x104fdc1257328f08184ba73131c53cae
                       e698e36119421149ea8c712456697d30
   HMAC-SHA-256-128: 0x104fdc1257328f08184ba73131c53cae
```

```
Test Case #3: HMAC-SHA-256 with 112-byte (multi-block) input
              and 32-byte key
Key_len         : 32
Key             : 0x0102030405060708090a0b0c0d0e0f10
                    1112131415161718191a1b1c1d1e1f20
Data_len        : 112
Data            : "abcdbcdecdefdefgefghfghighijhijk
                   ijkljklmklmnlmnomnopnopqabcdbcde
                   cdefdefgefghfghighijhijkijkljklm
                   klmnlmnomnopnopq"
HMAC-SHA-256    : 0x470305fc7e40fe34d3eeb3e773d95aab
                    73acf0fd060447a5eb4595bf33a9d1a3
HMAC-SHA-256-128: 0x470305fc7e40fe34d3eeb3e773d95aab

Test Case #4:  HMAC-SHA-256 with 8-byte input and 32-byte key
Key_len         : 32
Key             : 0x0b repeated 32 times
Data_len        : 8
Data            : 0x4869205468657265
Data            : "Hi There"
HMAC-SHA-256    : 0x198a607eb44bfbc69903a0f1cf2bbdc5
                    ba0aa3f3d9ae3c1c7a3b1696a0b68cf7
HMAC-SHA-256-128: 0x198a607eb44bfbc69903a0f1cf2bbdc5

Test Case #5:  HMAC-SHA-256 with 28-byte input and 4-byte key
Key_len         : 4
Key             : "Jefe"
Data_len        : 28
Data            : "what do ya want for nothing?"
HMAC-SHA-256    : 0x5bdcc146bf60754e6a042426089575c7
                    5a003f089d2739839dec58b964ec3843

Test Case #6: HMAC-SHA-256 with 50-byte input and 32-byte key
Key_len         : 32
Key             : 0xaa repeated 32 times
Data_len        : 50
Data            : 0xdd repeated 50 times
HMAC-SHA-256    : 0xcdcb1220d1ecccea91e53aba3092f962
                    e549fe6ce9ed7fdc43191fbde45c30b0
HMAC-SHA-256-128: 0xcdcb1220d1ecccea91e53aba3092f962

Test Case #7: HMAC-SHA-256 with 50-byte input and 37-byte key
Key_len         : 37
Key             : 0x0102030405060708090a0b0c0d0e0f10
                    1112131415161718191a1b1c1d1e1f20
                    2122232425
Data_len        : 50
Data            : 0xcd repeated 50 times
```

```
   HMAC-SHA-256    : 0xd4633c17f6fb8d744c66dee0f8f07455
                       6ec4af55ef07998541468eb49bd2e917

   Test Case #8: HMAC-SHA-256 with 20-byte input and 32-byte key
   Key_len         : 32
   Key             : 0x0c repeated 32 times
```

```
Data_len        : 20
Data            : "Test With Truncation"
HMAC-SHA-256    : 0x7546af01841fc09b1ab9c3749a5f1c17
                    d4f589668a587b2700a9c97c1193cf42
HMAC-SHA-256-128: 0x7546af01841fc09b1ab9c3749a5f1c17

Test Case #9: HMAC-SHA-256 with 54-byte input and 80-byte key
Key_len         : 80
Key             : 0xaa repeated 80 times
Data_len        : 54
Data            : "Test Using Larger Than Block-Size Key -
                   Hash Key First"
HMAC-SHA-256    : 0x6953025ed96f0c09f80a96f78e6538db
                    e2e7b820e3dd970e7ddd39091b32352f

Test Case #10: HMAC-SHA-256 with 73-byte (multi-block) input
               and 80-byte key
Key_len         : 80
Key             : 0xaa repeated 80 times
Data_len        : 73
Data            : "Test Using Larger Than Block-Size Key and
                   Larger Than One Block-Size Data"
HMAC-SHA-256    : 0x6355ac22e890d0a3c8481a5ca4825bc8
                    84d3e7a1ff98a2fc2ac7d8e064c3b2e6
```

4. IKE Interactions

4.1 Phase 1 Identifier

For Phase 1 negotiations, IANA has assigned a Hash Algorithm ID of 4 for SHA2-256.

For further information on the use of Hash Algorithm IDs within IKE, see [IKE].

4.2 Phase 2 Identifier

For Phase 2 negotiations, IANA has assigned an AH Transform Identifier of 5 for AH_SHA2-256.

For Phase 2 negotiations, IANA has assigned an AH/ESP Authentication Algorithm Attribute Value of 5 for HMAC-SHA2-256.

For further information on the use of Transform Identifiers and Attributes Value within IKE, see [IKE] and [DOI].

5. Security Considerations

The security provided by HMAC-SHA-256-128 is based upon the strength of SHA-256. At the time of this writing there are no practical cryptographic attacks against SHA-256.

As is true with any cryptographic algorithm, part of its strength lies in the correctness of the algorithm implementation, the security of the key management mechanism and its implementation, the strength of the associated secret key, and upon the correctness of the implementation in all of the participating systems. This draft contains test vectors to assist in verifying the correctness of HMAC-SHA-256-128 code.

6. IANA Considerations

IANA has assigned Hash Algorithm ID 4 to SHA2-256.
IANA has assigned AH Transform Identifier 5 to AH_SHA2-256.
IANA has assigned AH/ESP Authentication Algorithm Attribute Value 5 to HMAC-SHA2-256.

7. Intellectual Property Rights Statement

Pursuant to the provisions of [RFC-2026], the authors represent that they have disclosed the existence of any proprietary or intellectual property rights in the contribution that are reasonably and personally known to the authors. The authors do not represent that they personally know of all potentially pertinent proprietary and intellectual property rights owned or claimed by the organizations they represent or third parties.

The IETF takes no position regarding the validity or scope of any intellectual property 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; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication 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 Secretariat.

8. Acknowledgments

Portions of this text were unabashedly borrowed from [HMAC-SHA].

Thanks to Hugo Krawczyk for his comments and recommendations.

9. References

[AH] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[ARCH] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[DOI] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP,"

[ESP] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[HASH] Bellare, M., R. Canetti and H. Krawczyk, "Keying Hash Functions for Message Authentication," Advances in Cryptography, Crypto96 Proceedings, June 1996.

[HMAC] Krawczyk, H., M. Bellare and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104, February 1997.

[HMAC-SHA] Madson, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH," RFC 2404, November 1998.

[HMAC-TEST] Cheng, P. and R. Glenn, "Test Cases for HMAC-MD5 and HMAC-SHA-1", RFC 2202, September 1997.

[IKE] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC-2026] Bradner, S., "The Internet Standards Process -- Revision 3", RFC2026, October 1996.

[RFC-2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC-2119, March 1997.

[ROADMAP] Thayer, R., N. Doraswamy, and R. Glenn, "IP Security Document Roadmap", RFC 2411, November 1998.

[SHA2-1] NIST, Draft FIPS PUB 180-2 "Specifications for the Secure Hash Standard," May 2001. http://csrc.nist.gov/encryption/shs/dfips-180-2.pdf

[SHA2-2] "Descriptions of SHA-256, SHA-384, and SHA-512." http://csrc.nist.gov/cryptval/shs/sha256-384-512.pdf

10. Authors' Addresses

Sheila Frankel
NIST
820 West Diamond Ave.
Room 680
Gaithersburg, MD 20899
Phone: +1 (301) 975-3297

Email: sheila.frankel@nist.gov

Scott Kelly
Black Storm Networks
250 Cambridge Ave

Palo Alto CA 94304
Phone: +1 (650) 617-2934
Email: scott@bstormnetworks.com

The IPsec working group can be contacted through the chairs:

Barbara Fraser
Cisco Systems Inc.
Email: byfraser@cisco.com

Theodore Ts'o
Massachusetts Institute of Technology
Email: tytso@mit.edu

11. Full Copyright Statement