Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 9, 2007
S. Kelly
Aruba Networks
S. Frankel
NIST
January 5, 2007

Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 With IPsec
draft-kelly-ipsec-ciph-sha2-01

Status of this Memo

Copyright Notice

Abstract

This specification describes the use of HMAC in conjunction with the SHA-256, SHA-384, and SHA-512 algorithms in IPsec. These algorithms may be used as the basis for data origin authentication and integrity verification mechanisms for the AH, ESP, IKE and IKEv2 protocols, and also as Pseudo-Random Functions (PRFs) for IKE and IKEv2. Truncated output lengths are specified for the authentication-related variants, with the corresponding algorithms designated as HMAC-SHA-256-128,

HMAC-SHA-384-192, and HMAC-SHA-512-256. The PRF variants are not truncated, and are called HMAC-SHA-PRF-256, HMAC-SHA-PRF-384, and HMAC-SHA-PRF-512.

Table of Contents

1. Introduction

This document specifies the use of SHA-256, SHA-384, and SHA-512 [SHA2-1] combined with HMAC [HMAC] as data origin authentication and integrity verification mechanisms for the IPsec AH [AH], ESP [ESP], IKE [IKE], and IKEv2 [IKEv2] protocols. Output truncation is specified for these variants, with the corresponding algorithms designated as HMAC-SHA-256-128, HMAC-SHA-384-192, and HMAC-SHA-512-256. These truncation lengths are chosen in accordance with the birthday bound for each algorithm.

This specification also describes untruncated variants of these algorithms as PRFs for use with IKE and IKEv2, and those algorithms are called HMAC-SHA-PRF-256, HMAC-SHA-PRF-384, and HMAC-SHA-PRF-512. For ease of reference, these PRF algorithms and the authentication variants described above are collectively referred to below as "the HMAC-SHA-256+ algorithms."

The goal of the PRF variants is to provide secure pseudo-random functions suitable for generation of keying material and other protocol-specific numeric quantities, while the goal of the authentication variants is to ensure that packets are authentic and cannot be modified in transit. The relative security of HMAC-SHA-256+ when used in either case is dependent on the distribution scope and unpredictability of the associated secret key. If the key is unpredictable and known only by the sender and recipient, these algorithms ensure that only parties holding an identical key can derive the associated values.

2. The HMAC-SHA-256+ Algorithms

[SHA2-1] and [SHA2-2] describe the underlying SHA-256, SHA-384, and SHA-512 algorithms, while [HMAC] describes the HMAC algorithm. The HMAC algorithm provides a framework for inserting various hashing algorithms such as SHA-256, and [SHA256+] describes combined usage of

these algorithms. The following sections describe the various characteristics and requirements of the HMAC-SHA-256+ algorithms when used with IPsec.

2.1. Keying Material

Requirements for keying material vary depending on whether the algorithm is functioning as a PRF or as an authentication/integrity mechanism. In the case of authentication/integrity, key lengths are fixed according to the output length of the algorithm in use. In the case of PRFs, key lengths are variable, but guidance is given to ensure interoperability. These distinctions are described further

below.

Before describing key requirements for each usage, it is important to clarify some terms we use below:

Block size: the size of the data block the underlying hash algorithm operates upon; for SHA-256, this is 512 bits. For SHA-384 and SHA-512, this is 1024 bits.

Output length: the size of the hash value produced by the underlying hash algorithm. For SHA-256, this is 256 bits, for SHA-384 this is 384 bits, and for SHA-512, this is 512 bits.

Authenticator length: the size of the "authenticator" in bits. This only applies to authentication/integrity related algorithms, and refers to the bit length remaining after truncation. In this specification, this is always half the output length of the underlying hash algorithm.

2.1.1. Data Origin Authentication and Integrity Verification Usage

HMAC-SHA-256+ are secret key algorithms. While no fixed key length is specified in [HMAC], this specification requires that when used as an integrity/authentication algorithm, a fixed key length equal to the output length of the hash functions MUST be supported, and key lengths other than the output length of the associated hash function MUST NOT be supported.

These key length restrictions are based in part on the

recommendations in [HMAC] (key lengths less than the output length decrease security strength, and keys longer than the output length do not significantly increase security strength), and in part because allowing variable length keys for IPsec authenticator functions would create interoperability issues.

2.1.2. Pseudo-Random Function (PRF) Usage

IKE and IKEv2 use PRFs for generating keying material and for authentication of the IKE_SA. The IKEv2 specification differentiates between PRFs with fixed key sizes and those with variable key sizes, and so we give some special guidance for this below.

When a PRF described in this document is used with IKE or IKEv2, it is considered to have a variable key length, and keys are derived in the following ways (note that we simply reiterate that which is specified in [HMAC]):

- o If the length of the key is exactly the algorithm block size, use it as-is.

- o If the key is shorter than the block size, lengthen it to exactly the block size by padding it on the right with zero bits. However, note that [HMAC] strongly discourages a key length less than the output length. Nonetheless, we describe handling of shorter lengths here in recognition of shorter lengths typically chosen for IKE or IKEv2 preshared keys.

- o If the key is longer than the block size, shorten it by computing the corresponding hash algorithm output over the entire key value, and treat the resulting output value as your HMAC key. Note that this will always result in a key that is less than the block size in length, and this key value will therefore require 0-padding (as described above) prior to use.

2.1.3. Randomness and Key Strength

[HMAC] discusses requirements for key material, including a requirement for strong randomness. Therefore, a strong pseudo-random function MUST be used to generate the required key for use with HMAC-

SHA-256+. At the time of this writing there are no published weak keys for use with any HMAC-SHA-256+ algorithms.

2.1.4. Key Distribution

[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 and integrity verification, the key distribution mechanism must ensure that unique keys are allocated and that they are distributed only to the parties participating in the communication.

2.1.5. Refreshing Keys

There are no currently practical attacks against the algorithms recommended here, and especially against the key sizes recommended here. However, as noted in [HMAC] "...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."

Putting this into perspective, this specification requires 256, 384, or 512-bit keys produced by a strong PRF for use as a MAC. A brute force attack on such keys would take longer to mount than the universe has been in existence. On the other hand, weak keys (e.g. dictionary words) would be dramatically less resistant to attack. It

is important to take these points, along with the specific threat model for your particular application and the current state of the art with respect to attacks on SHA-256, SHA-384, and SHA-512 into account when determining an appropriate upper bound for HMAC key lifetimes

2.2. Padding

The HMAC-SHA-256 algorithms operate on 512-bit blocks of data, while the HMAC-SHA-384 and HMAC-SHA-512 algorithms operate on 1024-bit blocks of data. Padding requirements are specified in [SHA2-1] as part of the underlying SHA-256, SHA-384, and SHA-512 algorithms, so if you implement according to [SHA2-1], you do not need to add any additional padding as far as the HMAC-SHA-256+ algorithms specified here are concerned. With regard to "implicit packet padding" as

defined in [AH], no implicit packet padding is required.

2.3. Truncation

The HMAC-SHA-256+ algorithms each produce a nnn-bit value, where nnn corresponds to the output bit length of the algorithm, e.g. HMAC-SHA-nnn. For use as an authenticator, this nnn-bit value can be truncated as described in [HMAC]. When used as a data origin authentication and integrity verification algorithm in ESP, AH, IKE, or IKEv2, a truncated value using the first nnn/2 bits -- exactly half the algorithm output size -- MUST be supported. No other authenticator value lengths are supported by this specification.

Upon sending, the truncated value is stored within the authenticator field. Upon receipt, the entire nnn-bit value is computed and the first nnn/2 bits are compared to the value stored in the authenticator field, with the value of 'nnn' depending on the negotiated algorithm.

[HMAC] discusses potential security benefits resulting from truncation of the output MAC value, and in general, encourages HMAC users to perform MAC truncation. In the context of IPsec, a truncation length of nnn/2 bits is selected because it corresponds to the birthday attack bound for each of the HMAC-SHA-256+ algorithms, and it simultaneously serves to minimize the additional bits on the wire resulting from use of this facility.

2.4. Using HMAC-SHA-256+ As PRFs in IKE and IKEv2

The HMAC-SHA-PRF-256 algorithm is identical to HMAC-SHA-256-128, except that variable-length keys are permitted, and the truncation step is NOT performed. Likewise, the implementations of HMAC-SHA-PRF-384 and HMAC-SHA-PRF-512 are identical to those of HMAC-SHA-384-

192 and HMAC-SHA-512-256 respectively, except that again, truncation is NOT performed.

2.5. Interactions with the ESP, IKE, or IKEv2 Cipher Mechanisms

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

2.6. HMAC-SHA-256+ Parameter Summary

The following table serves to summarize the various quantities associated with the HMAC-SHA-256+ algorithms.

Algorithm ID	Block Size	Output Length	Trunc. Length	Key Length	Algorithm Type
HMAC-SHA-256-128	512	256	128	256	auth/integ
HMAC-SHA-384-192	1024	384	192	384	auth/integ
HMAC-SHA-512-256	1024	512	256	512	auth/integ
HMAC-SHA-256-PRF	512	256	(none)	variable	PRF
HMAC-SHA-384-PRF	1024	384	(none)	variable	PRF
HMAC-SHA-512-PRF	1024	512	(none)	variable	PRF

2.7. Test Vectors

The following test cases include the key, the data, and the resulting authenticator and/or PRF values for each algorithm. The values of keys and data are either ASCII character strings (surrounded by double quotes) or hexadecimal numbers. 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.

2.7.1. PRF Test Vectors

These test cases were borrowed from RFC 4231 [HMAC-TEST]. For reference implementations of the underlying hash algorithms, see [SHA256+]. Note that for testing purposes, PRF output is considered to be simply the untruncated algorithm output.

Test Case PRF-1:

```
   Key =          0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b
                  0b0b0b0b                          (20 bytes)

   Data =         4869205468657265                  ("Hi There")

   HMAC-SHA-256-PRF = b0344c61d8db38535ca8afceaf0bf12b
                      881dc200c9833da726e9376c2e32cff7

   HMAC-SHA-384-PRF = afd03944d84895626b0825f4ab46907f
                      15f9dadbe4101ec682aa034c7cebc59c
                      faea9ea9076ede7f4af152e8b2fa9cb6

   HMAC-SHA-512-PRF = 87aa7cdea5ef619d4ff0b4241a1d6cb0
                      2379f4e2ce4ec2787ad0b30545e17cde
                      daa833b7d6b8a702038b274eaea3f4e4
                      be9d914eeb61f1702e696c203a126854


Test Case PRF-2:
Key =          4a656665                          ("Jefe")

Data =         7768617420646f2079612077616e7420  ("what do ya want ")
               666f72206e6f7468696e673f          ("for nothing?")

HMAC-SHA-256-PRF = 5bdcc146bf60754e6a042426089575c7
                   5a003f089d2739839dec58b964ec3843

HMAC-SHA-384-PRF = af45d2e376484031617f78d2b58a6b1b
                   9c7ef464f5a01b47e42ec3736322445e
                   8e2240ca5e69e2c78b3239ecfab21649

HMAC-SHA-512-PRF = 164b7a7bfcf819e2e395fbe73b56e0a3
                   87bd64222e831fd610270cd7ea250554
                   9758bf75c05a994a6d034f65f8f0e6fd
                   caeab1a34d4a6b4b636e070a38bce737
```

```
Test Case PRF-3:
Key             aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                aaaaaaaa                          (20 bytes)

Data =          dddddddddddddddddddddddddddddddd
                dddddddddddddddddddddddddddddddd
                dddddddddddddddddddddddddddddddd
                dddd                              (50 bytes)

HMAC-SHA-256-PRF = 773ea91e36800e46854db8ebd09181a7
                   2959098b3ef8c122d9635514ced565fe

HMAC-SHA-384-PRF = 88062608d3e6ad8a0aa2ace014c8a86f
                   0aa635d947ac9febe83ef4e55966144b
                   2a5ab39dc13814b94e3ab6e101a34f27

HMAC-SHA-512-PRF = fa73b0089d56a284efb0f0756c890be9
                   b1b5dbdd8ee81a3655f83e33b2279d39
                   bf3e848279a722c806b485a47e67c807
                   b946a337bee8942674278859e13292fb


Test Case PRF-4:
Key =           0102030405060708090a0b0c0d0e0f10
                111213141516171819                (25 bytes)

Data =          cdcdcdcdcdcdcdcdcdcdcdcdcdcdcdcd
                cdcdcdcdcdcdcdcdcdcdcdcdcdcdcdcd
                cdcdcdcdcdcdcdcdcdcdcdcdcdcdcdcd
                cdcd                              (50 bytes)

HMAC-SHA-256-PRF = 82558a389a443c0ea4cc819899f2083a
                   85f0faa3e578f8077a2e3ff46729665b

HMAC-SHA-384-PRF = 3e8a69b7783c25851933ab6290af6ca7
                   7a9981480850009cc5577c6e1f573b4e
                   6801dd23c4a7d679ccf8a386c674cffb

HMAC-SHA-512-PRF = b0ba465637458c6990e5a8c5f61d4af7
                   e576d97ff94b872de76f8050361ee3db
                   a91ca5c11aa25eb4d679275cc5788063
                   a5f19741120c4f2de2adebeb10a298dd
```

```
   Test Case PRF-5:
   Key =           aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                   aaaaaa                            (131 bytes)

   Data =          54657374205573696e67204c61726765  ("Test Using Large")
                   72205468616e20426c6f636b2d53697a  ("r Than Block-Siz")
                   65204b6579202d2048617368204b6579  ("e Key - Hash Key")
                   204669727374                      (" First")


   HMAC-SHA-256-PRF = 60e431591ee0b67f0d8a26aacbf5b77f
                      8e0bc6213728c5140546040f0ee37f54

   HMAC-SHA-384-PRF = 4ece084485813e9088d2c63a041bc5b4
                      4f9ef1012a2b588f3cd11f05033ac4c6
                      0c2ef6ab4030fe8296248df163f44952

   HMAC-SHA-512-PRF = 80b24263c7c1a3ebb71493c1dd7be8b4
                      9b46d1f41b4aeec1121b013783f8f352
                      6b56d037e05f2598bd0fd2215d6a1e52
                      95e64f73f63f0aec8b915a985d786598
```

```
   Test Case PRF-6:

   Key =          aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                  aaaaaa                            (131 bytes)

   Data =         54686973206973206120746573742075  ("This is a test u")
                  73696e672061206c6172676572207468  ("sing a larger th")
                  616e20626c6f636b2d73697a65206b65  ("an block-size ke")
                  7920616e642061206c61726765722074  ("y and a larger t")
                  68616e20626c6f636b2d73697a652064  ("han block-size d")
                  6174612e20546865206b6579206e6565  ("ata. The key nee")
                  647320746f2062652068617368656420  ("ds to be hashed ")
                  6265666f7265206265696e6720757365  ("before being use")
                  642062792074686520484d414320616c  ("d by the HMAC al")
                  676f726974686d2e                  ("gorithm.")

   HMAC-SHA-256-PRF = 9b09ffa71b942fcb27635fbcd5b0e944
                      bfdc63644f0713938a7f51535c3a35e2

   HMAC-SHA-384-PRF = 6617178e941f020d351e2f254e8fd32c
                      602420feb0b8fb9adccebb82461e99c5
                      a678cc31e799176d3860e6110c46523e

   HMAC-SHA-512-PRF = e37b6a775dc87dbaa4dfa9f96e5e3ffd
                      debd71f8867289865df5a32d20cdc944
```

```
                   b6022cac3c4982b10d5eeb55c3e4de15
                   134676fb6de0446065c97440fa8c6a58
```

2.7.2. Authenticator Test Vectors

In the following sections are test cases for HMAC-SHA256-128, HMAC-SHA384-192, and HMAC-SHA512-256. PRF outputs are also included for convenience. These test cases were generated using the SHA256+ reference code provided in [SHA256+].

2.7.2.1. SHA256 Authentication Test Vectors


```
   Test Case AUTH256-1:
   Key =          0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b
                  0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b  (32 bytes)

   Data =         4869205468657265                  ("Hi There")

   HMAC-SHA-256-PRF = 198a607eb44bfbc69903a0f1cf2bbdc5
                      ba0aa3f3d9ae3c1c7a3b1696a0b68cf7

   HMAC-SHA-256-128 = 198a607eb44bfbc69903a0f1cf2bbdc5


Test Case AUTH256-2:
Key =          4a6566654a6566654a6566654a656665   ("JefeJefeJefeJefe")
               4a6566654a6566654a6566654a656665   ("JefeJefeJefeJefe")

Data =         7768617420646f2079612077616e7420  ("what do ya want ")
               666f72206e6f7468696e673f          ("for nothing?")

HMAC-SHA-256-PRF = 167f928588c5cc2eef8e3093caa0e87c
                   9ff566a14794aa61648d81621a2a40c6

HMAC-SHA-256-128 = 167f928588c5cc2eef8e3093caa0e87c
```

```
   Test Case AUTH256-3:
   Key =          aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  (32 bytes)

   Data =         dddddddddddddddddddddddddddddddd
                  dddddddddddddddddddddddddddddddd
                  dddddddddddddddddddddddddddddddd
                  dddd                              (50 bytes)

   HMAC-SHA-256-PRF = cdcb1220d1ecccea91e53aba3092f962
                      e549fe6ce9ed7fdc43191fbde45c30b0

   HMAC-SHA-256-128 = cdcb1220d1ecccea91e53aba3092f962
```



```
   Test Case AUTH256-4:
   Key =          0102030405060708090a0b0c0d0e0f10
                  1112131415161718191a1b1c1d1e1f20  (32 bytes)

   Data =         cdcdcdcdcdcdcdcdcdcdcdcdcdcdcdcd
                  cdcdcdcdcdcdcdcdcdcdcdcdcdcdcdcd
                  cdcdcdcdcdcdcdcdcdcdcdcdcdcdcdcd
                  cdcd                              (50 bytes)

   HMAC-SHA-256-PRF = 372efcf9b40b35c2115b1346903d2ef4
                      2fced46f0846e7257bb156d3d7b30d3f

   HMAC-SHA-256-128 = 372efcf9b40b35c2115b1346903d2ef4
```

2.7.2.2. SHA384 Authentication Test Vectors

```
   Test Case AUTH384-1:
```

```
   Key =          0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b
                  0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b
                  0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b  (48 bytes)

   Data =         4869205468657265                  ("Hi There")

   HMAC-SHA-384-PRF = b6a8d5636f5c6a7224f9977dcf7ee6c7
                      fb6d0c48cbdee9737a959796489bddbc
                      4c5df61d5b3297b4fb68dab9f1b582c2

   HMAC-SHA-384-128 = b6a8d5636f5c6a7224f9977dcf7ee6c7
                      fb6d0c48cbdee973


Test Case AUTH384-2:
Key =          4a6566654a6566654a6566654a656665   ("JefeJefeJefeJefe")
               4a6566654a6566654a6566654a656665   ("JefeJefeJefeJefe")
               4a6566654a6566654a6566654a656665   ("JefeJefeJefeJefe")

Data =         7768617420646f2079612077616e7420  ("what do ya want ")
               666f72206e6f7468696e673f          ("for nothing?")

HMAC-SHA-384-PRF = 2c7353974f1842fd66d53c452ca42122
                   b28c0b594cfb184da86a368e9b8e16f5
                   349524ca4e82400cbde0686d403371c9

HMAC-SHA-384-192 = 2c7353974f1842fd66d53c452ca42122
                   b28c0b594cfb184d
```



```
   Test Case AUTH384-3:
   Key =          aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  (48 bytes)

   Data =         dddddddddddddddddddddddddddddddd
                  dddddddddddddddddddddddddddddddd
                  dddddddddddddddddddddddddddddddd
                  dddd                              (50 bytes)

   HMAC-SHA-384-PRF = 809f439be00274321d4a538652164b53
```

```
                        554a508184a0c3160353e3428597003d
                        35914a18770f9443987054944b7c4b4a

   HMAC-SHA-384-192 = 809f439be00274321d4a538652164b53
                        554a508184a0c316



   Test Case AUTH384-4:
   Key =            0102030405060708090a0b0c0d0e0f10
                    1112131415161718191a1b1c1d1e1f20
                    0a0b0c0d0e0f10111213141516171819  (48 bytes)

   Data =           cdcdcdcdcdcdcdcdcdcdcdcdcdcdcdcd
                    cdcdcdcdcdcdcdcdcdcdcdcdcdcdcdcd
                    cdcdcdcdcdcdcdcdcdcdcdcdcdcdcdcd
                    cdcd                              (50 bytes)

   HMAC-SHA-384-PRF = 5b540085c6e6358096532b2493609ed1
                        cb298f774f87bb5c2ebf182c83cc7428
                        707fb92eab2536a5812258228bc96687

   HMAC-SHA-384-192 = 5b540085c6e6358096532b2493609ed1
                        cb298f774f87bb5c
```

2.7.2.3. SHA512 Authentication Test Vectors


```
   Test Case AUTH512-1:
   Key =            0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b
                    0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b
                    0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b
```

```
                   0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b  (64 bytes)

      Data =         4869205468657265                 ("Hi There")

      HMAC-SHA-512-PRF = 637edc6e01dce7e6742a99451aae82df
                         23da3e92439e590e43e761b33e910fb8
                         ac2878ebd5803f6f0b61dbce5e251ff8
                         789a4722c1be65aea45fd464e89f8f5b

      HMAC-SHA-512-256 = 637edc6e01dce7e6742a99451aae82df
                         23da3e92439e590e43e761b33e910fb8



   Test Case AUTH512-2:
   Key =          4a6566654a6566654a6566654a656665   ("JefeJefeJefeJefe")
                  4a6566654a6566654a6566654a656665   ("JefeJefeJefeJefe")
                  4a6566654a6566654a6566654a656665   ("JefeJefeJefeJefe")
                  4a6566654a6566654a6566654a656665   ("JefeJefeJefeJefe")

   Data =         7768617420646f2079612077616e7420  ("what do ya want ")
                  666f72206e6f7468696e673f          ("for nothing?")

   HMAC-SHA-512-PRF = cb370917ae8a7ce28cfd1d8f4705d614
                      1c173b2a9362c15df235dfb251b15454
                      6aa334ae9fb9afc2184932d8695e397b
                      fa0ffb93466cfcceaae38c833b7dba38

   HMAC-SHA-512-256 = cb370917ae8a7ce28cfd1d8f4705d614
                      1c173b2a9362c15df235dfb251b15454
```

```
Test Case AUTH512-3:
Key =          aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
               aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
               aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
               aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  (64 bytes)

Data =         dddddddddddddddddddddddddddddddd
               dddddddddddddddddddddddddddddddd
               dddddddddddddddddddddddddddddddd
               dddd                              (50 bytes)

HMAC-SHA-512-PRF = 2ee7acd783624ca9398710f3ee05ae41
                   b9f9b0510c87e49e586cc9bf961733d8
                   623c7b55cebefccf02d5581acc1c9d5f
                   b1ff68a1de45509fbe4da9a433922655

HMAC-SHA-512-256 = 2ee7acd783624ca9398710f3ee05ae41
                   b9f9b0510c87e49e586cc9bf961733d8


Test Case AUTH512-4:
Key =          0a0b0c0d0e0f10111213141516171819
               0102030405060708090a0b0c0d0e0f10
               1112131415161718191a1b1c1d1e1f20
               2122232425262728292a2b2c2d2e2f30
               3132333435363738393a3b3c3d3e3f40  (64 bytes)

Data =         cdcdcdcdcdcdcdcdcdcdcdcdcdcdcdcd
               cdcdcdcdcdcdcdcdcdcdcdcdcdcdcdcd
               cdcdcdcdcdcdcdcdcdcdcdcdcdcdcdcd
               cdcd                              (50 bytes)

HMAC-SHA-512-PRF = 5e6688e5a3daec826ca32eaea224eff5
                   e700628947470e13ad01302561bab108
                   b8c48cbc6b807dcfbd850521a685babc
                   7eae4a2a2e660dc0e86b931d65503fd2

HMAC-SHA-512-256 = 5e6688e5a3daec826ca32eaea224eff5
                   e700628947470e13ad01302561bab108
```

3. Security Considerations

In a general sense, the security provided by the HMAC-SHA-256+ algorithms is based both upon the strength of the underlying hash

algorithm, and upon the additional strength derived from the HMAC

construct. At the time of this writing there are no practical cryptographic attacks against SHA-256, SHA-384, SHA-512 or HMAC. However, as with any cryptographic algorithm, an important component of these algorithms' strength lies in the correctness of the algorithm implementation, the security of the key management mechanism, the strength of the associated secret key, and upon the correctness of the implementation in all of the participating systems. This specification contains test vectors to assist in verifying the correctness of the algorithm implementation, but these in no way verify the correctness (or security) of the surrounding security infrastructure.

3.1. HMAC Key Length vs Truncation Length

There are important differences between the security levels afforded by HMAC-SHA1-96 and the HMAC-SHA-256+ algorithms, but there are also considerations which are somewhat counter-intuitive. There are two different axes along which we gauge the security of these algorithms: HMAC output length and HMAC key length. If we assume the HMAC key is a well-guarded secret which can only be determined through offline attacks on observed values, and that its length is less than or equal to the output length of the underlying hash algorithm, then the key's strength is directly proportional to its length. And if we assume an adversary has no knowledge of the HMAC key, then the probability of guessing a correct MAC value for any given packet is directly proportional to the HMAC output length.

This specification defines truncation to output lengths of either 128 192, or 256 bits. It is important to note that at this time, it is not clear that HMAC-SHA-256 with a truncation length of 128 bits is any more secure than HMAC-SHA1 with the same truncation length, assuming the adversary has no knowledge of the HMAC key. This is because in such cases, the adversary must predict only those bits which remain after truncation. Since in both cases that output length is the same (128 bits), the adversary's odds of correctly guessing the value are also the same in either case: 1 in 2^128. Again, if we assume the HMAC key remains unknown to the attacker, then only a bias in one of the algorithms would distinguish one from the other. Currently, no such bias is known to exist in either HMAC-SHA1 or HMAC-SHA-256+.

If, on the other hand, the attacker is focused on guessing the HMAC key, and we assume that the hash algorithms are indistinguishable when viewed as PRF's, then the HMAC key length provides a direct measure of the underlying security: the longer the key, the harder it is to guess. This means that with respect to passive attacks on the HMAC key, size matters - and the HMAC-SHA-256+ algorithms provide more security in this regard than HMAC-SHA1-96.

4. IANA Considerations

This document does not specify the conventions for using SHA256+ for IKE Phase 1 negotiations. For IKE Phase 2 negotiations, IANA has assigned the following authentication algorithm identifiers:

HMAC-SHA2-256: 5

HMAC-SHA2-384: 6

HMAC-SHA2-512: 7

For use of HMAC-SHA-256+ as a PRF in IKEv2, IANA has assigned the following IKEv2 Pseudo-random function (type 2) transform identifiers:

PRF_HMAC_SHA2_256 [TBA-1]

PRF_HMAC_SHA2_384 [TBA-2]

PRF_HMAC_SHA2_512 [TBA-3]

For the use of HMAC-SHA-256+ algorithms for data origin authentication and integrity verification in IKEv2, ESP or AH, IANA has assigned the following IKEv2 integrity (type 3) transform identifiers:

AUTH_HMAC_SHA2_256_128 [TBA-4]

AUTH_HMAC_SHA2_384_192 [TBA-5]

AUTH_HMAC_SHA2_512_256 [TBA-6]

5. Acknowledgements

Portions of this text were unabashedly borrowed from [HMAC-SHA1], and from [HMAC-TEST]. Thanks to Hugo Krawczyk for comments and recommendations on early revisions of this document, and thanks also to Russ Housley and Steve Bellovin for various security-related comments and recommendations.

6. Normative References

[AH] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[ARCH] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[ESP] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

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

[HMAC-SHA1] Madsen, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, November 1998.

[HMAC-TEST] Nystrom, M., "Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512", RFC 4231, December 2005.

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

[IKEv2] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[SHA2-1] NIST, "FIPS PUB 180-2 'Specifications for the Secure Hash

Standard'", 2004 FEB, <http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf>.

[SHA2-2] NIST, "Descriptions of SHA-256, SHA-384, and SHA-512", 2001 MAY, <http://csrc.nist.gov/cryptval/shs/sha256-384-512.pdf>.

[SHA256+] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, July 2006.

Authors' Addresses

Scott G. Kelly
Aruba Networks
1322 Crossman Ave
Sunnyvale, CA 94089
US

Email: scott@hyperthought.com

Sheila Frankel
NIST
Bldg. 222 Room B264
Gaithersburg, MD 20899
US

Email: sheila.frankel@nist.gov

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).