# ARCHIVED PUBLICATION

The attached publication,

FIPS Publication 198
(dated March 6, 2002),

was superseded on July 29, 2008 and is provided here only for historical purposes.

For the most current revision of this publication, see:
http://csrc.nist.gov/publications/PubsFIPS.html#198-1.

# FIPS PUB 198

## FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION

# The Keyed-Hash Message Authentication Code (HMAC)

**CATEGORY: COMPUTER SECURITY** **SUBCATEGORY: CRYPTOGRAPHY**

Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8900

Issued March 6, 2002


**U.S. Department of Commerce**
Donald L. Evans, Secretary

**Technology Administration**
Philip J. Bond, Under Secretary

**National Institute of Standards and Technology**
Arden L. Bement, Jr., Director

## Foreword

The Federal Information Processing Standards Publication Series of the National Institute of Standards and Technology (NIST) is the official series of publications relating to standards and guidelines adopted and promulgated under the provisions of Section 5131 of the Information Technology Management Reform Act of 1996 (Public Law 104-106) and the Computer Security Act of 1987 (Public Law 100-235). These mandates have given the Secretary of Commerce and NIST important responsibilities for improving the utilization and management of computer and related telecommunications systems in the Federal government. The NIST, through its Information Technology Laboratory, provides leadership, technical guidance, and coordination of government efforts in the development of standards and guidelines in these areas.

Comments concerning Federal Information Processing Standards Publications are welcomed and should be addressed to the Director, Information Technology Laboratory, National Institute of Standards and Technology, 100 Bureau Drive, Stop 8900, Gaithersburg, MD 20899-8900.

William Mehuron, Director
Information Technology
Laboratory

---

## Abstract

This standard describes a keyed-hash message authentication code (HMAC), a mechanism for message authentication using cryptographic hash functions. HMAC can be used with any iterative Approved cryptographic hash function, in combination with a shared secret key. The cryptographic strength of HMAC depends on the properties of the underlying hash function. The HMAC specification in this standard is a generalization of Internet RFC 2104, *HMAC, Keyed-Hashing for Message Authentication*, and ANSI X9.71, *Keyed Hash Message Authentication Code.*

*Keywords*: computer security, cryptography, HMAC, MAC, message authentication, Federal Information Processing Standard (FIPS).

**Federal Information Processing Standards Publication 198**

**2002 March 6**

**Announcing the Standard for**

# The Keyed-Hash Message Authentication Code (HMAC)

Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology (NIST) after approval by the Secretary of Commerce pursuant to Section 5131 of the Information Technology Management Reform Act of 1996 (Public Law 104-106) and the Computer Security Act of 1987 (Public Law 100-235).

1. **Name of Standard.** Keyed-Hash Message Authentication Code (HMAC) (FIPS PUB 198).

2. **Category of Standard.** Computer Security Standard. **Subcategory.** Cryptography.

**3. Explanation.** This standard specifies an algorithm for applications requiring message authentication. Message authentication is achieved via the construction of a message authentication code (MAC). MACs based on cryptographic hash functions are known as HMACs.

The purpose of a MAC is to authenticate both the source of a message and its integrity without the use of any additional mechanisms. HMACs have two functionally distinct parameters, a message input and a secret key known only to the message originator and intended receiver(s). Additional applications of keyed-hash functions include their use in challenge-response identification protocols for computing responses, which are a function of both a secret key and a challenge message.

An HMAC function is used by the message sender to produce a value (the MAC) that is formed by condensing the secret key and the message input. The MAC is typically sent to the message receiver along with the message. The receiver computes the MAC on the received message using the same key and HMAC function as was used by the sender, and compares the result computed with the received MAC. If the two values match, the message has been correctly received, and the receiver is assured that the sender is a member of the community of users that share the key.

The HMAC specification in this standard is a generalization of HMAC as specified in Internet RFC 2104, *HMAC, Keyed-Hashing for Message Authentication*, and ANSI X9.71, *Keyed Hash Message Authentication Code.*

4. **Approving Authority.** Secretary of Commerce.

**5. Maintenance Agency.** Department of Commerce, National Institute of Standards and Technology, Information Technology Laboratory (ITL).

**6. Applicability.** This standard is applicable to all Federal departments and agencies for the protection of sensitive unclassified information that is not subject to section 2315 of Title 10, United States Code, or section 3502(2) of Title 44, United States Code. This standard shall be used in designing, acquiring and implementing keyed-hash message authentication techniques in systems that Federal departments and agencies operate or which are operated for them under contract. The adoption and use of this standard is available on a voluntary basis to private and commercial organizations.

**7. Specifications.** Federal Information Processing Standard (FIPS) 198, Keyed-Hash Message Authentication Code (HMAC) (affixed).

**8. Implementations.** The authentication mechanism described in this standard may be implemented in software, firmware, hardware, or any combination thereof. NIST has developed a Cryptographic Module Validation Program that will test implementations for conformance with this HMAC standard. Information on this program is available at http://csrc.nist.gov/cryptval/.

Agencies are advised that keys used for HMAC applications should not be used for other purposes.

**9. Other Approved Security Functions.** HMAC implementations that comply with this standard shall employ cryptographic algorithms, cryptographic key generation algorithms and key management techniques that have been approved for protecting Federal government sensitive information. Approved cryptographic algorithms and techniques include those that are either:

a. specified in a Federal Information Processing Standard (FIPS),
b. adopted in a FIPS or NIST Recommendation and specified either in an appendix to the FIPS or NIST Recommendation or in a document referenced by the FIPS or NIST Recommendation, or
c. specified in the list of Approved security functions for FIPS 140-2.

**10. Export Control.** Certain cryptographic devices and technical data regarding them are subject to Federal export controls. Exports of cryptographic modules implementing this standard and technical data regarding them must comply with these Federal regulations and be licensed by the Bureau of Export Administration of the U.S. Department of Commerce. Applicable Federal government export controls are specified in Title 15, Code of Federal Regulations (CFR) Part 740.17; Title 15, CFR Part 742; and Title 15, CFR Part 774, Category 5, Part 2.

**11. Implementation Schedule.** This standard becomes effective on September 6, 2002.

**12. Qualifications.** The security afforded by the HMAC function is dependent on maintaining the secrecy of the key. Therefore, users must guard against disclosure of these keys. While it is the intent of this standard to specify a mechanism to provide message authentication, conformance to this standard does not assure that a particular implementation is secure. It is the responsibility of the implementer to ensure that any module containing an HMAC implementation is designed and built in a secure manner.

Similarly, the use of a product containing an implementation that conforms to this standard does not guarantee the security of the overall system in which the product is used. The responsible authority in each agency shall assure that an overall system provides an acceptable level of security.

Since a standard of this nature must be flexible enough to adapt to advancements and innovations in science and technology, this standard will be reviewed every five years in order to assess its adequacy.

**13. Waiver Procedure.** Under certain exceptional circumstances, the heads of Federal agencies, or their delegates, may approve waivers to Federal Information Processing Standards (FIPS). The heads of such agencies may redelegate such authority only to a senior official designated pursuant to Section 3506(b) of Title 44, U.S. Code. Waivers shall be granted only when compliance with this standard would

a. adversely affect the accomplishment of the mission of an operator of Federal computer system or

b. cause a major adverse financial impact on the operator that is not offset by government-wide savings.

Agency heads may act upon a written waiver request containing the information detailed above. Agency heads may also act without a written waiver request when they determine that conditions for meeting the standard cannot be met. Agency heads may approve waivers only by a written decision that explains the basis on which the agency head made the required finding(s). A copy of each such decision, with procurement sensitive or classified portions clearly identified, shall be sent to: National Institute of Standards and Technology; ATTN: FIPS Waiver Decision, Information Technology Laboratory, 100 Bureau Drive, Stop 8900, Gaithersburg, MD 20899-8900.

In addition, notice of each waiver granted and each delegation of authority to approve waivers shall be sent promptly to the Committee on Government Operations of the House of Representatives and the Committee on Government Affairs of the Senate and shall be published promptly in the Federal Register.

When the determination on a waiver applies to the procurement of equipment and/or services, a notice of the waiver determination must be published in the Commerce Business Daily as a part of the notice of solicitation for offers of an acquisition or, if the waiver determination is made after that notice is published, by amendment to such notice.

A copy of the waiver, any supporting documents, the document approving the waiver and any supporting and accompanying documents, with such deletions as the agency is authorized and decides to make under Section 552(b) of Title 5, U.S. Code, shall be part of the procurement documentation and retained by the agency.

**14. Where to obtain copies.** This publication is available by accessing http://csrc.nist.gov/publications/. A list of other available computer security publications, including ordering information, can be obtained from NIST Publications List 91, which is available at the same web site. Alternatively, copies of NIST computer security publications are available from: National Technical Information Service (NTIS), 5285 Port Royal Road, Springfield, VA 22161.

**Federal Information Processing Standards Publication 198**

**2002 March 6**

**Specifications for**

# The Keyed-Hash Message Authentication Code

**TABLE OF CONTENTS**

# 1. INTRODUCTION

Providing a way to check the integrity of information transmitted over or stored in an unreliable medium is a prime necessity in the world of open computing and communications. Mechanisms that provide such integrity checks based on a secret key are usually called message authentication codes (MACs). Typically, message authentication codes are used between two parties that share a secret key in order to authenticate information transmitted between these parties. This standard defines a MAC that uses a cryptographic hash function in conjunction with a secret key. This mechanism is called HMAC and is a generalization of HMAC as specified in [1] and [3].

HMAC shall be used in combination with an Approved cryptographic hash function. HMAC uses a secret key for the calculation and verification of the MACs. The main goals behind the HMAC construction [3] are:

- To use available hash functions without modifications; in particular, hash functions that perform well in software, and for which code is freely and widely available,
- To preserve the original performance of the hash function without incurring a significant degradation,
- To use and handle keys in a simple way,
- To have a well-understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions on the underlying hash function, and
- To allow for easy replaceability of the underlying hash function in the event that faster or more secure hash functions are later available.

# 2. GLOSSARY OF TERMS AND ACRONYMS

## 2.1 Glossary of Terms

The following definitions are used throughout this standard:

*Approved*: FIPS-approved or NIST recommended. An algorithm or technique that is either 1) specified in a FIPS or NIST Recommendation, or 2) adopted in a FIPS or NIST Recommendation and specified either the FIPS or NIST Recommendation, or in a document referenced by the FIPS or NIST Recommendation.

*Cryptographic key (key)*: a parameter used in conjunction with a cryptographic algorithm that determines the specific operation of that algorithm. In this standard, the cryptographic key is used by the HMAC algorithm to produce a MAC on the data.

*Hash function*: an Approved mathematical function that maps a string of arbitrary length (up to a pre-determined maximum size) to a fixed length string. It may be used to produce a checksum, called a hash value or message digest, for a potentially long string or message.

*Keyed-hash based message authentication code (HMAC)*: a message authentication code that uses a cryptographic key in conjunction with a hash function.

*Message Authentication Code (MAC):* a cryptographic checksum that results from passing data through a message authentication algorithm. In this standard, the message authentication algorithm is called HMAC, while the result of applying HMAC is called the MAC.

*Secret key*: a cryptographic key that is uniquely associated with one or more entities. The use of the term "secret" in this context does not imply a classification level; rather the term implies the need to protect the key from disclosure or substitution.

## 2.2 Acronyms

The following acronyms and abbreviations are used throughout this standard:

| | |
|---|---|
| FIPS | Federal Information Processing Standard |
| FIPS PUB | FIPS Publication |
| HMAC | Keyed-Hash Message Authentication Code |
| MAC | Message Authentication Code |
| NIST | National Institute of Standards and Technology |

### 2.3 HMAC Parameters and Symbols

HMAC uses the following parameters:

| | |
|---|---|
| *B* | Block size (in bytes) of the input to the Approved hash function. |
| *H* | An Approved hash function. |
| *ipad* | Inner pad; the byte x’36’ repeated *B* times. |
| *K* | Secret key shared between the originator and the intended receiver(s). |

$K_0$ The key $K$ after any necessary pre-processing to form a $B$ byte key.

$L$ Block size (in bytes) of the output of the Approved hash function.

*opad* Outer pad; the byte x'5c' repeated $B$ times.

$t$ The number of bytes of MAC.

*text* The data on which the HMAC is calculated; *text* does **not** include the padded key. The length of *text* is $n$ bits, where $0 \leq n < 2^B - 8B$.

x'*N*' Hexadecimal notation, where each symbol in the string '*N*' represents 4 binary bits.

|| Concatenation

⊕ Exclusive-Or operation.

## 3. CRYPTOGRAPHIC KEYS

The size of the key, $K$, shall be equal to or greater than $L/2$, where $L$ is the size of the hash function output. Note that keys greater than $L$ bytes do not significantly increase the function strength. Applications that use keys longer than $B$-bytes shall first hash the key using $H$ and then use the resultant $L$-byte string as the HMAC key, $K$. Keys shall be chosen at random using an Approved key generation method and shall be changed periodically. Note that the keys should be protected in a manner that is consistent with the value of the data that is to be protected (i.e., the *text* that is authenticated using the HMAC function).

## 4. TRUNCATED OUTPUT

A well-known practice with MACs is to truncate their output (i.e., the length of the MAC used is less than the length of the output of the MAC function $L$). Applications of this standard may truncate the output of HMAC. When a truncated HMAC is used, the $t$ leftmost bytes of the HMAC computation shall be used as the MAC. The output length, $t$, shall be no less than four bytes (i.e., $4 \leq t \leq L$). However, $t$ shall be at least $\frac{L}{2}$ bytes (i.e., $\frac{L}{2} \leq t \leq L$) unless an application or protocol makes numerous trials impractical. For example, a low bandwidth channel might prevent numerous trials on a 4 byte MAC, or a protocol might allow only a small number of invalid MAC attempts. See Appendix B.

## 5. HMAC SPECIFICATION

To compute a MAC over the data '*text*' using the HMAC function, the following operation is performed:

$$\boldsymbol{MAC(text)_t = HMAC(K, text)_t = H((K_0 \oplus opad) \| H((K_0 \oplus ipad) \| text))_t}$$

Table 1 illustrates the step by step process in the HMAC algorithm, which is depicted in Figure 1.

**Table 1: The HMAC Algorithm**

| STEPS | STEP-BY-STEP DESCRIPTION |
|---|---|
| *Step 1* | If the length of $K = B$: set $K_0 = K$. Go to step 4. |
| *Step 2* | If the length of $K > B$: hash $K$ to obtain an $L$ byte string, then append $(B\text{-}L)$ zeros to create a B-byte string $K_0$ (i.e., $K_0 = H(K) \| 00...00$). Go to step 4. |
| *Step 3* | If the length of $K < B$: append zeros to the end of $K$ to create a $B$-byte string $K_0$ (e.g., if $K$ is 20 bytes in length and $B = 64$, then $K$ will be appended with 44 zero bytes 0x00). |
| *Step 4* | Exclusive-Or $K_0$ with *ipad* to produce a $B$-byte string: $\boldsymbol{K_0 \oplus ipad}$. |
| *Step 5* | Append the stream of data '*text*' to the string resulting from step 4: $\boldsymbol{(K_0 \oplus ipad) \| text}$. |
| *Step 6* | Apply $H$ to the stream generated in step 5: $\boldsymbol{H((K_0 \oplus ipad) \| text)}$. |
| *Step 7* | Exclusive-Or $K_0$ with *opad*: $\boldsymbol{K_0 \oplus opad}$. |
| *Step 8* | Append the result from step 6 to step 7: $\boldsymbol{(K_0 \oplus opad) \| H((K_0 \oplus ipad) \| text)}$. |
| *Step 9* | Apply $H$ to the result from step 8: $\boldsymbol{H((K_0 \oplus opad) \| H((K_0 \oplus ipad) \| text))}$. |
| *Step 10* | Select the leftmost $t$ bytes of the result of step 9 as the MAC. |

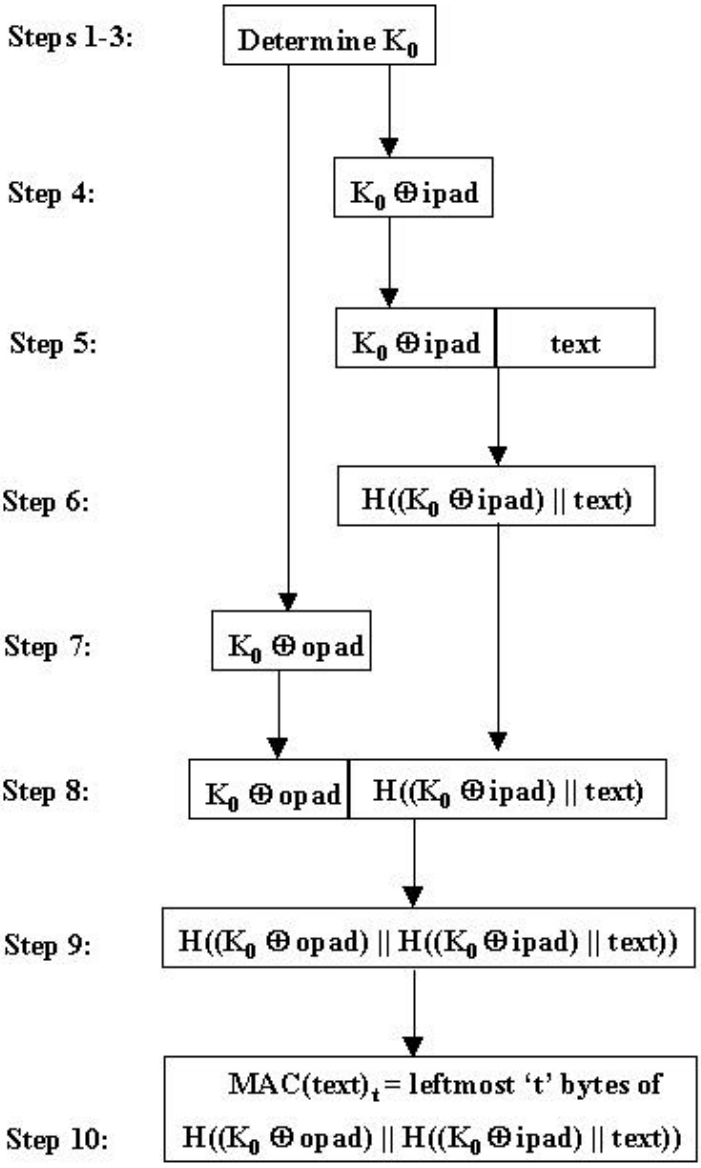


**Figure 1: Illustration of the HMAC Construction**

## 6. IMPLEMENTATION NOTE

The HMAC algorithm is specified for an arbitrary Approved cryptographic hash function, *H*. With minor modifications, an HMAC implementation can easily replace one hash function, *H*, with another hash function, *H'*.

Conceptually, the intermediate results of the compression function on the *B*-byte blocks ($K_0 \oplus ipad$) and ($K_0 \oplus opad$) can be precomputed once, at the time of generation of the

key *K*, or before its first use. These intermediate results can be stored and then used to initialize *H* each time that a message needs to be authenticated using the same key. For each authenticated message using the key *K*, this method saves the application of the hash function of *H* on two *B*-byte blocks (i.e., on ($K \oplus ipad$) and ($K \oplus opad$)). This saving may be significant when authenticating short streams of data. **These stored intermediate values shall be treated and protected in the same manner as secret keys.**

Choosing to implement HMAC in this manner has no effect on interoperability.

Object identifiers (OIDs) for HMAC are posted at http://csrc.nist.gov/csor, along with procedures for adding new OIDs.

## APPENDIX A: HMAC EXAMPLES

These examples are provided in order to promote correct implementations of HMAC. The SHA-1 hash function used in these examples is specified in [4].

### A.1 SHA-1 with 64-Byte Key

Text: "Sample #1"

```
Key:  00010203  04050607  08090a0b  0c0d0e0f
      10111213  14151617  18191a1b  1c1d1e1f
      20212223  24252627  28292a2b  2c2d2e2f
      30313233  34353637  38393a3b  3c3d3e3f

K0:   00010203  04050607  08090a0b  0c0d0e0f
      10111213  14151617  18191a1b  1c1d1e1f
      20212223  24252627  28292a2b  2c2d2e2f
      30313233  34353637  38393a3b  3c3d3e3f
```

$K_0 \oplus$ ipad:

```
      36373435  32333031  3e3f3c3d  3a3b3839
      26272425  22232021  2e2f2c2d  2a2b2829
      16171415  12131011  1e1f1c1d  1a1b1819
      06070405  02030001  0e0f0c0d  0a0b0809
```

(Key $\oplus$ ipad)||text:

```
      36373435  32333031  3e3f3c3d  3a3b3839
      26272425  22232021  2e2f2c2d  2a2b2829
      16171415  12131011  1e1f1c1d  1a1b1819
      06070405  02030001  0e0f0c0d  0a0b0809
      53616d70  6c652023  31
```

Hash((Key $\oplus$ ipad)||text):

```
      bcc2c68c  abbbf1c3  f5b05d8e  7e73a4d2
      7b7e1b20
```

$K_0 \oplus$ opad:

```
      5c5d5e5f  58595a5b  54555657  50515253
      4c4d4e4f  48494a4b  44454647  40414243
      7c7d7e7f  78797a7b  74757677  70717273
      6c6d6e6f 68696a6b   64656667  60616263
```

($K_0 \oplus$ opad) || Hash((Key $\oplus$ ipad)||text):

```
      5c5d5e5f  58595a5b  54555657  50515253
      4c4d4e4f  48494a4b  44454647  40414243
```

```
7c7d7e7f 78797a7b   74757677  70717273
6c6d6e6f  68696a6b  64656667  60616263
bcc2c68c  abbbf1c3  f5b05d8e  7e73a4d2
7b7e1b20
```

HMAC(Key, Text) = Hash(($K_0$ ⊕ opad) || Hash((Key ⊕ ipad)||text)):

```
4f4ca3d5  d68ba7cc  0a1208c9  c61e9c5d
a0403c0a
```

20-byte HMAC(Key, Text):

```
4f4ca3d5  d68ba7cc  0a1208c9  c61e9c5d
a0403c0a
```

### A.2 SHA-1 with 20-Byte Key

Text: "Sample #2"

Key:
```
30313233  34353637  38393a3b  3c3d3e3f
40414243
```

$K_0$:
```
30313233  34353637  38393a3b  3c3d3e3f
40414243  00000000  00000000  00000000
00000000  00000000  00000000  00000000
00000000  00000000  00000000  00000000
```

$K_0$ ⊕ ipad:
```
06070405  02030001  0e0f0c0d  0a0b0809
76777475  36363636  36363636  36363636
36363636  36363636  36363636  36363636
36363636  36363636  36363636  36363636
```

(Key ⊕ ipad)||text:
```
06070405  02030001  0e0f0c0d  0a0b0809
76777475  36363636  36363636  36363636
36363636  36363636  36363636  36363636
36363636  36363636  36363636  36363636
53616d70  6c652023  32800000  00000000
00000000  00000000  00000000  00000000
00000000  00000000  00000000  00000000
00000000  00000000  00000000  00000248
```

Hash((Key ⊕ ipad)||text):
```
74766e5f  6913e8cb  6f7f108a  11298b15
010c353a
```

$K_0 \oplus$ opad:

```
6c6d6e6f  68696a6b  64656667  60616263
1c1d1e1f  5c5c5c5c  5c5c5c5c  5c5c5c5c
5c5c5c5c  5c5c5c5c  5c5c5c5c  5c5c5c5c
5c5c5c5c  5c5c5c5c  5c5c5c5c  5c5c5c5c
```

($K_0 \oplus$ opad) || Hash(($K_0 \oplus$ ipad)||text):

```
6c6d6e6f  68696a6b  64656667  60616263
1c1d1e1f  5c5c5c5c  5c5c5c5c  5c5c5c5c
5c5c5c5c  5c5c5c5c  5c5c5c5c  5c5c5c5c
5c5c5c5c  5c5c5c5c  5c5c5c5c  5c5c5c5c
74766e5f  6913e8cb  6f7f108a  11298b15
010c353a
```

HMAC(Key, Text) = Hash(($K_0 \oplus$ opad) || Hash(($K_0 \oplus$ ipad)||text)):

```
0922d340  5faa3d19  4f82a458  30737d5c
c6c75d24
```

20-byte HMAC(Key, Text):

```
0922d340  5faa3d19  4f82a458  30737d5c
c6c75d24
```

### A.3 SHA-1 with 100-Byte Key

Text: "Sample #3"

Key:

```
50515253  54555657  58595a5b  5c5d5e5f
60616263  64656667  68696a6b  6c6d6e6f
70717273  74757677  78797a7b  7c7d7e7f
80818283  84858687  88898a8b  8c8d8e8f
90919293  94959697  98999a9b  9c9d9e9f
a0a1a2a3  a4a5a6a7  a8a9aaab  acadaeaf
b0b1b2b3
```

Hash(Key):

```
a4aabe16  54e78da4  40d2a403  015636bf
4bb2f329
```

$K_0$:

```
a4aabe16  54e78da4  40d2a403  015636bf
4bb2f329  00000000  00000000  00000000
00000000  00000000  00000000  00000000
00000000  00000000  00000000  00000000
```

$K_0 \oplus$ ipad:

```
929c8820  62d1bb92   76e49235  37600089
7d84c51f  36363636   36363636  36363636
36363636  36363636   36363636  36363636
36363636  36363636   36363636  36363636
```

(Key ⊕ ipad)||text:

```
929c8820  62d1bb92  76e49235  37600089
7d84c51f  36363636  36363636  36363636
36363636  36363636  36363636  36363636
36363636  36363636  36363636  36363636
53616d70  6c652023  33
```

Hash((Key ⊕ ipad)||text):

```
d98315c4  2152bea0  d057de97  84427676
2a1a5576
```

$K_0$ ⊕ opad:

```
f8f6e24a  08bbd1f8  1c8ef85f  5d0a6ae3
17eeaf75  5c5c5c5c  5c5c5c5c  5c5c5c5c
5c5c5c5c  5c5c5c5c  5c5c5c5c  5c5c5c5c
5c5c5c5c  5c5c5c5c  5c5c5c5c  5c5c5c5c
```

($K_0$ ⊕ opad) || Hash((Key ⊕ ipad)||text):

```
f8f6e24a  08bbd1f8  1c8ef85f  5d0a6ae3
17eeaf75  5c5c5c5c  5c5c5c5c  5c5c5c5c
5c5c5c5c  5c5c5c5c  5c5c5c5c  5c5c5c5c
5c5c5c5c  5c5c5c5c  5c5c5c5c  5c5c5c5c
d98315c4  2152bea0  d057de97  84427676
2a1a5576
```

HMAC(Key, Text) = Hash(($K_0$ ⊕ opad) || Hash((Key ⊕ ipad)||text)):

```
bcf41eab  8bb2d802  f3d05caf  7cb092ec
f8d1a3aa
```

20-byte HMAC(Key, Text):

```
bcf41eab  8bb2d802  f3d05caf  7cb092ec
f8d1a3aa
```

### A.4 SHA-1 with 49-Byte Key, Truncated to 12-Byte HMAC

Text: "Sample #4"

Key:

```
70717273  74757677  78797a7b  7c7d7e7f
80818283  84858687  88898a8b  8c8d8e8f
90919293  94959697  98999a9b  9c9d9e9f
```

```
      a0

K0:   70717273  74757677  78797a7b  7c7d7e7f
      80818283  84858687  88898a8b  8c8d8e8f
      90919293  94959697  98999a9b  9c9d9e9f
      a0000000  00000000  00000000  00000000

K0 ⊕ ipad:
      46474445  42434041  4e4f4c4d  4a4b4849
      b6b7b4b5  b2b3b0b1  bebfbcbd  babbb8b9
      a6a7a4a5  a2a3a0a1  aeafacad  aaaba8a9
      96363636  36363636  36363636  36363636

(Key ⊕ ipad)||text:
      46474445  42434041  4e4f4c4d  4a4b4849
      b6b7b4b5  b2b3b0b1  bebfbcbd  babbb8b9
      a6a7a4a5  a2a3a0a1  aeafacad  aaaba8a9
      96363636  36363636  36363636  36363636
      53616d70  6c652023  34

Hash((Key ⊕ ipad)||text):
      bf1e889d  876c34b7  bef3496e  d998c8d1
      16673a2e

K0 ⊕ opad:
      2c2d2e2f  28292a2b  24252627  20212223
      dcdddedf  d8d9dadb  d4d5d6d7  d0d1d2d3
      cccdcecf  c8c9cacb  c4c5c6c7  c0c1c2c3
      fc5c5c5c  5c5c5c5c  5c5c5c5c  5c5c5c5c

(K0 ⊕ opad) || Hash((Key ⊕ ipad)||text):
      2c2d2e2f  28292a2b  24252627  20212223
      dcdddedf  d8d9dadb  d4d5d6d7  d0d1d2d3
      cccdcecf  c8c9cacb  c4c5c6c7  c0c1c2c3
      fc5c5c5c  5c5c5c5c  5c5c5c5c  5c5c5c5c
      bf1e889d  876c34b7  bef3496e  d998c8d1
      16673a2e

HMAC(Key, Text) = Hash((K0 ⊕ opad) || Hash((Key ⊕ ipad)||text)):
      9ea886ef  e268dbec  ce420c75  24df32e0
      751a2a26

12-byte HMAC(Key, Text):
      9ea886ef  e268dbec  ce420c75
```

## APPENDIX B: A LIMITATION OF MAC ALGORITHMS

The successful verification of a MAC does not completely guarantee that the accompanying message is authentic: there is a chance that a source with no knowledge of the key can present a purported MAC on the plaintext message that will pass the verification procedure. For example, an arbitrary purported MAC of $t$ bits on an arbitrary plaintext message may be successfully verified with an expected probability of $(1/2)^t$. This limitation is inherent in any MAC algorithm.

The limitation is magnified if an application permits a given non-nauthentic message to be repeatedly presented for verification with different purported MACs. Each individual trial succeeds only with a small probability, $(1/2)^t$; however, for repeated trials, the probability increases that, eventually, one of the MACs will be successfully verified. Similarly, if an application permits a given purported MAC to be presented with different non-authentic messages, then the probability increases that, eventually, the MAC will be successfully verified for one of the messages.

Therefore, in general, if the MAC is truncated, then its length, $t$, should be chosen as large as is practical, with at least half as many bits as the output block size, $L$. The minimum value for $t$ is relaxed to 32 bits for applications in which the two types of repeated trials that are described in the previous paragraph are sufficiently restricted. For example, the application, or the protocol that controls the application, may monitor all of the plaintext messages and MACs that are presented for verification, and permanently reject any plaintext message or any MAC that is included in too many unsuccessful trials. Another example occurs when the bandwidth of the communications channel is low enough to preclude too many trials, of either type. In both cases, the maximum number of allowed unsuccessful trails must be pre-determined based on the risks associated with the sensitivity of the data, the length of $t$ and the MAC algorithm used.

## APPENDIX C: REFERENCES


[1] American Bankers Association, *Keyed Hash Message Authentication Code*, ANSI X9.71, Washington, D.C., 2000.

[2] National Institute of Standards and Technology, *Security Requirements for Cryptographic Modules*, Federal Information Processing Standards Publication 140-2, May 25, 2001.

[3] H. Krawczyk, M. Bellare, and R. Canetti, *HMAC: Keyed-Hashing for Message Authentication*, Internet Engineering Task Force, Request for Comments (RFC) 2104, February 1997.

[4] National Institute of Standards and Technology, *Secure Hash Standard (SHS)*, Federal Information processing Standards Publication 180-1, 17 April 1995.