The attached DRAFT document (provided here for HISTORICAL purposes) has been superseded by the following publication:

Publication Number: **Special Publication 800-107 Revision 1**

Title: **Recommendation for Applications Using Approved Hash Algorithms**

Publication Date: **08/24/2012**

- Final Publication: http://csrc.nist.gov/publications/nistpubs/800-107-rev1/sp800-107-rev1.pdf
- Related Information on CSRC: http://csrc.nist.gov/publications/PubsSPs.html#800-107
- Information on other NIST Computer Security Division publications and programs can be found at: http://csrc.nist.gov/

The following information was posted with the attached DRAFT document:

Special Publication 800-107 Revision 1, Recommendation for Using Approved Hash Algorithms
August 24, 2012

NIST announces the release of Special Publication 800-107, Revision 1, Recommendation for Using Approved Hash Algorithms. In this revision, the security properties of SHA-512/224 and SHA-512/256 are addressed, the discussion of the security of HMAC values has been expanded, the Randomized Hashing for Digital Signature discussion has been revised, and the Hash-based Key Derivation Function section has been rewritten to incorporate the “extraction-then-expansion” key derivation procedure specified in SP 800-56C and to discuss different approved hash-based key derivation functions.

Draft NIST Special Publication 800-107 (Revised)

Recommendation for Applications Using Approved Hash Algorithms

Quynh Dang

Computer Security Division
Information Technology Laboratory

C O M P U T E R S E C U R I T Y

September 2011

U.S. Department of Commerce
Rebecca Blank, Acting Secretary

National Institute of Standards and Technology
Patrick Gallagher, Under Secretary of Commerce for Standards and Technology

Abstract

Cryptographic hash functions that compute a fixed- length message digest from arbitrary length messages are widely used for many purposes in information security. This document provides security guidelines for achieving the required or desired security strengths when using cryptographic applications that employ the approved cryptographic hash functions specified in Federal Information Processing Standard (FIPS) 180-4. These include functions such as digital signature applications, Keyed-hash Message Authentication Codes (HMACs) and Hash-based Key Derivation Functions (Hash-based KDFs).

KEY WORDS: digital signatures, hash algorithms, cryptographic hash function, hash function, hash-based key derivation algorithms, hash value, HMAC, message digest, randomized hashing, random number generation, SHA, truncated hash values.

Acknowledgements

The author, Quynh Dang of the National Institute of Standards and Technology (NIST) gratefully appreciates the contributions and comments from Elaine Barker, William E. Burr, Shu-jen Chang, Lily Chen, Donna F. Dodson, Morris Dworkin, John Kelsey, Ray Perlner, W. Timothy Polk and Andrew Regenscheid. The author also appreciates comments from Daniel Brown, Hugo Krawczyk, Praveen Gauravaram and many other people at various Federal Agencies during the development of this Recommendation.

Table of Contents

Recommendation for Applications Using Approved Hash Algorithms

1 Introduction

A hash algorithm is used to map a message of arbitrary length to a fixed-length message digest. Federal Information Processing Standard (FIPS) 180-4, the Secure Hash Standard (SHS) [FIPS 180-4], specifies seven approved hash algorithms: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256. Secure hash algorithms are typically used with other cryptographic algorithms.

This Recommendation provides security guidelines for achieving the required or desired security strengths of several cryptographic applications that employ the approved cryptographic hash functions specified in FIPS 180-4, such as digital signature applications specified in FIPS 186-3 [FIPS 186-3], Keyed-hash Message Authentication Codes (HMACs) specified in FIPS 198-1 [FIPS 198-1] and Hash-based Key Derivation Functions specified in SP 800-56A [SP 800-56A] and SP 800-56B [SP 800-56B]. While the use of hash functions in HMAC-based key derivation functions is specified in SP 800-56C [SP 800-56C] and SP 800-108 [SP 800-108], these documents sufficiently address the security aspects of their use, so discussions of SP 800-56C and SP 800-108 are not included herein.

2 Authority

This Recommendation has been developed by the National Institute of Standards and Technology (NIST) in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets, but such standards and guidelines **shall not** apply to national security systems. This recommendation is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), Securing Agency Information Systems, as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided in A-130, Appendix III.

This Recommendation has been prepared for use by Federal agencies. It may be used by non-governmental organizations on a voluntary basis and is not subject to copyright (attribution would be appreciated by NIST).

Nothing in this Recommendation should be taken to contradict standards and guidelines made mandatory and binding on Federal agencies by the Secretary of Commerce under statutory authority. Nor should this Recommendation be interpreted as altering or

superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other federal official.

Conformance testing for implementations of this Recommendation will be conducted within the framework of the Cryptographic Algorithm Validation Program (CAVP) and the Cryptographic Module Validation Program (CMVP). The requirements of this Recommendation are indicated by the word "**shall**". Some of these requirements may be out-of-scope for CAVP and CMVP validation testing, and thus are the responsibility of entities using, implementing, installing, or configuring applications that incorporate this Recommendation.

3 Glossary of Terms, Acronyms and Mathematical Symbols

3.1 Terms and Definitions

Adversary	An entity that is not authorized to access or modify information, or who works to defeat any protections afforded the information.
Algorithm	A clearly-specified mathematical process for computation; a set of rules that, if followed, will give a prescribed result.
Approved	FIPS-approved and/or NIST-recommended. An algorithm or technique that is either 1) specified in a FIPS or NIST Recommendation, 2) adopted in a FIPS or NIST Recommendation or 3) specified in a list of NIST-approved security functions.
Approved hash algorithms	Hash algorithms specified in FIPS 180.
Bit string	An ordered sequence of 0 and 1 bits. The leftmost bit is the most significant bit of the string. The rightmost bit is the least significant bit of the string.
Bits of security	See security strength.
Block cipher	An invertible symmetric key cryptographic algorithm that operates on fixed-length blocks of input using a secret key and an unvarying transformation algorithm. The resulting output block is the same length as the input block.
Collision	An event in which two different messages have the same message digest.
Collision resistance	An expected property of a cryptographic hash function whereby it is computationally infeasible to find a collision, See "Collision".

Cryptographic hash function	A function that maps a bit string of arbitrary length to a fixed-length bit string and is expected to have the following three properties: 1. Collision resistance (see Collision resistance), 2. Preimage resistance (see Preimage resistance) and 3. Second preimage resistance (see Second preimage resistance). Approved cryptographic hash functions are specified in FIPS 180.
Digital signature	The result of applying two cryptographic functions (a cryptographic hash function, followed by a digital signature function, see FIPS 186-3 for details) to data that, when the functions are properly implemented, provides origin authentication, data integrity and signatory non-repudiation.
Hash algorithm	See cryptographic hash function. "Hash algorithm" and "cryptographic hash function" are used interchangeably in this Recommendation.
Hash output	See "message digest".
Hash value	See "message digest".
Key	A parameter used with a cryptographic algorithm that determines its operation in such a way that an entity with knowledge of the key can reproduce or reverse the operation, while an entity without knowledge of the key cannot. Examples applicable to this Recommendation include: 1. The computation of a keyed-hash message authentication code. 2. The verification of a keyed-hash message authentication code. 3. The generation of a digital signature on a message. 4. The verification of a digital signature.
Key Derivation Key	A key used as an input to a key derivation function to derive other keys.
Keying Material	A binary string, such that any non-overlapping segments of the string with the required lengths can be used as symmetric cryptographic keys and secret parameters, such as initialization vectors.
MAC algorithm	An algorithm that computes a MAC from a message and a key.
Message digest	The result of applying a cryptographic hash function to a message. Also known as a "hash value" or "hash output".

Preimage	A message *X* that produces a given message digest when it is processed by a hash function.
Preimage resistance	An expected property of a cryptographic hash function such that, given a randomly chosen message digest, *message_digest*, it is computationally infeasible to find a preimage of the *message_digest*, See "Preimage".
Random bit	A binary bit for which an attacker has exactly a 50% probability of success of guessing the value of the bit as either a zero or one.
Random bit generator	A device or algorithm that can produce a sequence of random bits that appears to be statistically independent and unbiased.
Randomized hashing	A process by which the input to a cryptographic hash function is randomized before being processed by the cryptographic hash function.
Random number	A value in a set that has an equal probability of being selected from the total population of possibilities and, hence, is unpredictable. A random number is an instance of an unbiased random variable, that is, the output produced by a uniformly distributed random process.
Second preimage	A message *X'*, that is different from a given message *X* , such that its message digest is the same as the known message digest of *X*.
Second preimage resistance	An expected property of a cryptographic hash function whereby it is computationally infeasible to find a second preimage of a known message digest, See "Second preimage".
Secret keying material	The binary data that is used to form secret keys, such as AES encryption or HMAC keys.
Security strength of a cryptographic algorithm or system	A number associated with the amount of work (that is, the number of operations) that is required to break a cryptographic algorithm or system. Security strength is measured in bits. If 2^N execution operations of the algorithm (or system) are required to break the cryptographic algorithm, then the security strength is *N* bits.
Security Strength of a secret key (or value) (in binary bits)	The required amount of work to find the key that is associated with some specific algorithm.
Shall	Used to indicate a requirement of this Recommendation.
Shared secret	A secret value that has been computed using a key agreement algorithm and is used as input to a key derivation function.

3.2 Acronyms

FIPS	Federal Information Processing Standard
SHA	Secure Hash algorithm
KDF	Key Derivation Function
MAC	Message Authentication Code
HMAC	Keyed-hash Message Authentication Code
RBG	Random Bit Generator

3.3 Symbols

K	HMAC key.
L	Length in bits of the full message digests from a hash function.
MacTag	Transmitted full or truncated HMAC output.
$\min(x, y)$	The minimum of x and y. For example, if $x < y$, then $\min(x, y) = x$.
λ	Length in bits of a *MacTag*.
$\lvert x \rvert$	The length (in bits) of the bit string x. For example, $\lvert 01100100 \rvert = 8$.

4 Approved Hash Algorithms

Currently, there are seven approved hash algorithms specified in FIPS 180-4: SHA-1, SHA-224, SHA-256, SHA-384 SHA-512, SHA-512/224 and SHA-512/256. These hash algorithms produce outputs of 160, 224, 256, 384, 512, 224 and 256 bits, respectively. The output of a hash algorithm is commonly known as a message digest, a hash value or a hash output.

4.1 Hash Function Properties

A cryptographic hash function[1] is expected to have the following three properties:

1. Collision resistance: It is computationally infeasible to find two different inputs to the cryptographic hash function that have the same hash value. That is, if *hash* is a cryptographic hash function, it is computationally infeasible to find two different inputs x and x' for which $hash(x) = hash(x')$. Collision resistance is measured by the amount of work that would be needed to find a collision for a cryptographic hash function with high probability. If the amount of work is 2^N, then the collision resistance is N bits. The estimated strength for collision resistance provided by a hash-function is half the length of the hash value produced by a given cryptographic hash function, i.e., the estimated security strength for collision

[1] The terms "cryptographic hash function" and "hash algorithm" are used interchangeably, depending on the context of the discussions throughout this Recommendation.

resistance is $L/2$ bits. For example, SHA-256 produces a (full-length) hash value of 256 bits; SHA-256 provides an estimated collision resistance of 128 bits.

2. Preimage resistance[2]: Given a randomly chosen hash value, *hash_value*, it is computationally infeasible to find an x so that *hash(x) = hash_value*. This property is also called the one-way property. Preimage resistance is measured by the amount of work that would be needed to find a preimage for a cryptographic hash function with high probability. If the amount of work is 2^N, then the preimage resistance is N bits. The estimated strength for preimage resistance provided by a hash-function is the length of the hash value produced by a given cryptographic hash function, i.e., the estimated security strength for preimage resistance is L bits. For example, SHA-256 produces a (full-length) hash value of 256 bits; SHA-256 provides an estimated preimage resistance of 256 bits.
3. Second preimage resistance: It is computationally infeasible to find a second input that has the same hash value as any other specified input. That is, given an input x, it is computationally infeasible to find a second input x' that is different from x, such that $hash(x) = hash(x')$. Second preimage resistance is measured by the amount of work that would be needed to find a second preimage for a cryptographic hash function with high probability; more detail can be found in Appendix A. If the amount of work is 2^N, then the second preimage resistance is N bits. The estimated strength for second preimage resistance provided by a hash-function is the length of the hash value produced by a given cryptographic hash function, i.e., the estimated security strength for second preimage reistance is L bits. For example, SHA-256 produces a (full-length) hash value of 256 bits; SHA-256 provides an estimated second preimage resistance of 256 bits.

The security strength of a cryptographic hash function is determined by either: its collision resistance strength, preimage resistance strength or second preimage resistance strength, depending on the property(ies) that the cryptographic application needs from the cryptographic hash function. If an application requires more than one property from the cryptographic hash function, then the weakest property is the security strength of the cryptographic hash function for the application. For instance, the security strength of a cryptographic hash function for digital signatures is defined as its collision resistance strength, because digital signatures require collision resistance and second preimage resistance from the cryptographic hash function, and the collision resistance strength of the cryptographic hash function ($L/2$) is less than its second preimage resistance strength (i.e., L).

A cryptographic hash function that is not suitable for one application might be suitable for other cryptographic applications that do not require the same security properties. For example, SHA-1 is not suitable for digital signature applications (as specified in FIPS 186-3) that require 112 bits of security unless randomized hashing is used as discussed in Section 5.2.3. However, SHA-1 can be used to provide 112 bits of security for HMAC

[2] There are slightly different definitions of preimage resistance of cryptographic hash functions in the literature.

applications (as specified in FIPS 198-1). In the case of digital signatures, SHA-1 does not provide the 112 bits of collision resistance needed to achieve the security strength. On the other hand, SHA-1 does provide the 112 bits of preimage resistance that is needed to achieve the security strength for HMAC. The security strengths of the approved cryptographic hash functions for different applications can be found in SP 800-57_Part 1 [SP 800-57].

4.2 Strengths of the Approved Hash Algorithms

Table 1 provides a summary of the security strengths for the hash function properties (discussed in the previous section) of the approved hash functions.

	SHA-1	**SHA-224**	**SHA-256**	**SHA-384**	**SHA-512**	**SHA-512/224**	**SHA-512/256**
Collision Resistance Strength in bits	< 80	112	128	192	256	112	128
Preimage Resistance Strength in bits	160	224	256	384	512	224	256
Second Preimage Resistance Strength in bits	105-160	201-224	201-256	384	394-512	224	256

Table 1: **Strengths of the Security Properties of the Approved Hash Algorithms**

As mentioned in Section 4.1, the estimated collision resistance strength of one of the approved cryptographic hash functions is, in general, estimated to be half the length of its hash value. This is currently believed to be true for all the approved hash functions except SHA-1. However, the latest cryptanalytic results for SHA-1 [SHA1 Attack] indicate that it may have a collision resistance strength that is considerably less than its expected strength of 80 bits.

The estimated preimage resistance strengths of the approved hash functions are provided in the above table. At the time that this Recommendation was written, there had been no

known short cuts to find the preimages of the hash values generated from the approved hash algorithms.

Except for SHA-384, SHA-512/224 and SHA-512/256, the second preimage resistance strengths of the approved cryptographic hash functions depend not only on the functions themselves, but also on the sizes of the messages that the cryptographic hash functions process [Second Preimage Attack]. In Table 1, the low end of each range applies to the situation where the message input length to the cryptographic hash function is the maximum length allowed by the hash function, while the high end of the range applies to the situation where the message input length is relatively small. Information on determining the actual second preimage resistance strengths of the approved cryptographic hash functions for different message lengths is provided in Appendix A. In the case of SHA-384, SHA-512/224 or SHA-512/256, the second preimage resistance strength does not depend on the message length; details can be found in Appendix A.

Note that the preimage resistance and the second preimage resistance strengths of any approved hash algorithm specified in FIPS 180-4 are stronger than its collision resistance.

5 Cryptographic Hash Function Usage

5.1 Truncated Message Digest

Some applications may require a value that is shorter than the (full-length) message digest provided by an approved cryptographic hash function specified in FIPS 180-4. In such cases, it may be appropriate to use a subset of the bits produced by the cryptographic hash function as the (shortened) message digest.

Let the (shortened) message digest be called a truncated message digest, and let λ be its desired length in bits. A truncated message digest may be used if the following requirements are met:

1. If collision resistance is required, λ **shall** be at least twice the required collision resistance strength s (in bits) for the truncated message digest (i.e., $\lambda \geq 2s$).
2. The length of the output block of the approved cryptographic hash function to be used **shall** be greater than λ (i.e., $L > \lambda$).
3. The λ left-most bits of the full-length message digest **shall** be selected as the truncated message digest.

   For example, if a truncated message digest of 96 bits is desired, the SHA-256 cryptographic hash function could be used (e.g., because it is available to the application, and provides an output larger than 96 bits). The leftmost 96 bits of the 256-bit message digest generated by SHA-256 are selected as the truncated message digest, and the rightmost 160 bits of the message digest are discarded.

For application interoperability, the standard method for truncating cryptographic hash function outputs is provided above strictly as a convenience for implementers and application developers. The proper use of a (shortened) message digest is an application-level issue.

Truncating the message digest can impact the security of an application. By truncating a message digest, the estimated collision resistance strength is reduced from $L/2$ to $\lambda/2$ (in bits). For the example in item 3 above, even though SHA-256 provides 128 bits of collision resistance, the collision resistance provided by the 96-bit truncated message digest is half the length of the truncated message digest, which is 48 bits, in this case.

The truncated message digest of λ bits provides an estimated preimage resistance of λ bits, not L bits, regardless of the cryptographic hash function used.

The estimated second preimage resistance strength of a message digest truncated to λ bits is determined as specified in Appendix A. For example, a 130-bit truncated message digest generated using SHA-256 has an estimated second preimage strength of 130 bits, rather than a value in the range specified in Table 1 above for SHA-256.

Truncating the message digest can have other impacts, as well. For example, applications that use a truncated message digest risk attacks based on confusion between different parties about the specific amount of truncation used, as well as the specific cryptographic hash function that was used to produce the truncated message digest. Any application using a truncated message digest is responsible for ensuring that the truncation amount and the cryptographic hash function used are known to all parties, with no chance of ambiguity. It is also important to note that there is no guarantee that truncation will not make any truncated message digest weaker than its expected security strength.

5.2 Digital Signatures

A cryptographic hash function is used to map a message of arbitrary length to a fixed-length message digest. For digital signature generation, this message digest is then signed by a digital signature algorithm. The resulting digital signature is used to verify who signed the message and whether it is the same message that was signed (e.g., whether there has been any accidental or deliberate alteration of the received message).

When two different messages have the same message digest (i.e., a collision is found), then a digital signature of one message may be used as a digital signature for the other message. If this happens, then a verified digital signature does not guarantee the authenticity of the signed message, because either one of the two messages could be considered as valid. Therefore, a cryptographic hash function used for digital signatures requires collision resistance. The approved cryptographic hash functions are believed to provide the collision resistance strengths as specified in the Table 1 of Section 4.1.

For digital signature applications, the security strength of a hash function without any preprocessing is generally its collision resistance strength. When appropriate processing is applied to the data before the hash value is computed, the security strength may be more than the collision resistance strength (see Section 5.2.3).

Without any preprocessing of the message input to the cryptographic hash function, the security strength of any digital signature that is generated using an algorithm specified in FIPS 186-3 is the minimum of the collision resistance strength of the hash algorithm and the security strength provided by the signing algorithm and key size. More information can be found in SP 800-57_Part 1. For instance, if a digital signature that is generated by one of the approved digital signature algorithms with SHA-1 as the cryptographic hash

function and key sizes specified in FIPS 186-3, then the security strength of this digital signature is less than 80 bits (see Table 1 in Section 4.1). Therefore, SHA-1 **shall not** be used in any new digital signature applications that require at least 80 bits of security strength. Furthermore, SHA-1 **shall not** be used for the generation of digital signatures after the end of 2013 (see SP 800-131A [SP 800-131A] for information about the use of key lengths for digital signature applications). More information on the security strengths of digital signature applications using the approved hash algorithms and the recommended lifetimes of cryptographic algorithm usage can be found in SP 800-57_Part 1.

There are several ways to use cryptographic hash functions with digital signature algorithms as described below.

5.2.1 Full-length Message Digests

Full-length message digests, as specified in FIPS 180-4, can be used with the approved digital signature algorithms, as specified in FIPS 186-3. The estimated security strength of collision resistance is provided in Table 1 above.

5.2.2 Truncated Message Digests

Truncated message digests may be used in generating digital signatures. However, the security of the cryptographic hash functions now depends on the lengths of the truncated message digests, as well as the cryptographic hash function that is used.

The length of truncated message digests used **shall** be at least twice the desired security strength required for the digital signature. For example, if a security strength of 112 bits is required, a truncated message digest of at least 224 bits must be produced. All hash functions except SHA-1 could be used to generate a 224-bit message digest, although, in the case of SHA-224 and SHA-512/224, the hash-value would not be truncated. It is recommended that the hash function chosen should minimize the number of truncation operations required to achieve the desired length for the hash value. For the example above, SHA-512/224 should be chosen over SHA-384 and SHA-512/256; all three hash algorithms require truncation to achieve the advertized length for the hash function (i.e., 224, 384 and 256 bits, respectively; see FIPS 180-4). However, to achieve a 224-bit message digest, SHA-384 and SHA-512/256 would require additional truncation.

5.2.3 Randomized Hashing for Digital Signatures

As described in Section 5.2, the security strength of a digital signature is limited by the collision resistance strength of the cryptographic hash function. However, when using the randomized hashing technique specified in SP 800-106 [SP 800-106], the security strength of the randomized cryptographic hash function is the minimum of its second preimage resistance strength and the total strength of its collision resistance strength plus the strength of the random value (i.e., *security strength* = min(*second preimage resistance strength*, (*collision resistance strength* + *random value strength*)).

When randomized hashing is used, the random value **shall** be generated with at least 112 bits of security strength and **shall** be at least 112 bits in length.

As stated in Section 4.1, SHA-1 has an estimated collision resistance strength that is less than 80 bits. Therefore, SHA-1 may not be suitable for digital signature applications that require 80 bits of security unless the randomized hashing technique is used. When SHA-1 is used with the randomized hashing technique specified in SP 800-106, the security strength provided is estimated at 160 bits (i.e., *security strength* = min(160, (80+112)); note that in this case, the collision resistance strength of SHA-1 can be considered to be 80 bits in the above formula, since the randomizing disallows the cryptanalytic attacks on SHA-1 that reduce its collision resistance strength. Therefore, SHA-1 will be suitable for applications requiring 112 or 128 bits of security when the randomized hashing technique specified in SP 800-106 is used.

5.3 Keyed-Hash Message Authentication Codes (HMAC)

5.3.1 Description

Message authentication codes (MACs) provide data authentication and integrity protection. Two types of algorithms for computing a MAC have been approved: 1) MAC algorithms that are based on approved block cipher algorithms (more information can be found in [SP 800-38B]) and 2) MAC algorithms that are based on cryptographic hash functions, called HMAC algorithms that are specified in FIPS 198-1. This section discusses the use of HMAC.

An output from an HMAC algorithm is called an HMAC output. The HMAC output is either used in its entirety, or is truncated (see Section 5.3.3) when it is transmitted for subsequent verification. The transmitted value is called a *MacTag*. The HMAC algorithm requires the use of a secret key that is shared between the entity that generates the HMAC output (e.g., a message sender), and the entity (or entities) that need to verify the transmitted *MacTag* (message receiver(s)).

The HMAC output is generated from a secret key and the string of "text" to be MACed (e.g., a message to be sent) using the HMAC algorithm. The *MacTag* is provided to the *MacTag* verifier, along with the "text" that was MACed (e.g., the sender transmits both the *MacTag* and the "text" that was MACed to the intended receiver).

The verifier computes an HMAC output on the received "text" using the same key and HMAC algorithm that was used to generate the *MacTag*, generates a (new) *MacTag* (either a full or truncated HMAC output), and then compares the generated *MacTag* with the received *MacTag*. If the two values match, the "text" has been correctly received, and the verifier is assured that the entity that generated the *MacTag* is a member of the community of users that share the key.

The security strength provided by the HMAC algorithm is determined by the security strength of the HMAC key and the length of the HMAC output.

5.3.2 The HMAC Key

The security strength of the HMAC algorithm depends, in part, on the security strength of the HMAC key, *K*. An HMAC key **shall** have a security strength that meets or exceeds the security strength required to protect the data over which the HMAC is computed.

The HMAC key **shall** be kept secret. When the secrecy of the HMAC key, *K*, is not preserved, an adversary that knows *K*, may impersonate any of the users that share that key in order to generate *MacTags* that seem to be authentic (i.e., *MacTags* that can be verified and are subsequently presumed to be authentic).

HMAC keys **shall** be one of the following:

1) A random bit string generated using an approved random bit generator as specified in SP 800-90 [SP 800-90],
2) A key established using an approved key establishment method as specified in SP 800-56A or SP 800-56B,
3) A pre-shared key (e.g., a key that has been manually distributed), or
4) A key produced using an approved key derivation function and a secret key derivation key. See Section 5.4 for more details on approved key derivation functions.

5.3.3 Truncation of HMAC Output

When applications truncate the HMAC outputs to generate *MacTags* to a desired length, λ, the λ left-most bits of the HMAC outputs **shall** be used as the *MacTags*. However, the output length, λ, **shall** be no less than 32 bits. For example, a low bandwidth channel or a desired high efficiency computation application such as audio or video casting application might use 32-bit *MacTags*.

5.3.4 Security Effect of the HMAC Key

The security strength effect of the HMAC key[3] is the minimum of the security strength of *K* and the value of 2*L* (i.e., *security strength* = min(*security strength of K*, 2*L*)). For example, if the security strength of *K* is 128 bits, and SHA-1 is used, then the security strength effect of the HMAC key is 128 bits, since for SHA-1, 2*L*= 320. However, when the security strength of *K* is greater than 320 bits, the security strength effect of the HMAC key is 320 bits. Therefore, it is not sensible to generate *K* with more than 2*L* bits of security strength.

The HMAC key *K* **shall** have a security strength that meets or exceeds the desired security strength of the HMAC application. For example, if the desired security strength of the HMAC application is 256 bits, the HMAC key *K* **shall** have a security strength of at least 256 bits.

---

[3] As described in [BCK1], the success of a collision attack on any approved HMAC algorithm in FIPS 198-1 that uses SHA-1 would require the collection of at least 2^{80} pairs of chosen plaintexts and their corresponding HMAC values. This is an impractical task. So, the collision attack is not considered in this document. The strength of the HMAC key here is the amount of work required for an attacker who performs a brute-force attack to discover the HMAC key *K* or $H(K0 \oplus opad)$ and $H(K0 \oplus ipad)$ in the HMAC construction (see FIPS 198-1) in order to generate authentic *MacTags* at any time.

5.3.5 Security of the HMAC Values

The successful verification of a *MacTag* does not completely guarantee that the accompanying text is authentic; there is a slight chance that an adversary with no knowledge of the HMAC key, *K*, can present a (*MacTag*, *text*) pair that will pass the verification procedure. From the perspective of an adversary that does not know the HMAC key *K* (i.e., the adversary is not among the community of users that share the key), the assurance of authenticity provided by a *MacTag* depends on its length and on the number of failed *MacTag* verifications allowed by a system for each value of the HMAC key.

Let 2^t be the number of failed MAC verifications allowed by a system for each value of the HMAC key. The *MacTag* length and the value of *t* need to be chosen to avoid an unacceptable probability of falsely accepting forged data. The likelihood of accepting forged data as authentic is $(1/2)^{(|MagTag| - t)}$. For example, if a *MacTag* length of 32 bits is used, and a system allows 2^{12} failed MAC verifications for any given value of the HMAC key, then the likelihood of accepting forged data is $(1/2)^{20}$. In order to increase assurance of authenticity, either the *MacTag* length would need to be increased, or the number of allowed failed MAC verifications for each value of the HMAC key would need to be decreased. To avoid having an unacceptable probability of falsely accepting forged data at any time, the value of HMAC key must be changed to a new value before number of failed MAC authentications reaches the maximum allowed number (2^t). For the example, above, the HMAC key must be changed before the number of failed MAC verifications reaches 2^{12} when the length of *MacTags* is 32 bits.

The table below provides the likelihoods of accepting forged data for different *MacTag* lengths and allowed numbers of MAC verifications using a given value of the HMAC key. This table is intended to assist the implementers of HMAC applications in security-sensitive systems in assessing the security risk associated with using *MacTags*.

MacTag Length	Number of allowed failed MAC verifications (2^t)		
	2^{20}	2^{30}	2^{35}
40	$(1/2)^{20}$	$(1/2)^{10}$	$(1/2)^{5}$
64	$(1/2)^{44}$	$(1/2)^{34}$	$(1/2)^{29}$
96	$(1/2)^{76}$	$(1/2)^{66}$	$(1/2)^{61}$

Table 2: Risks/Likelihoods of Accepting Forged Data

The $(1/2)^x$ entries in the cells are the risks of accepting forged data with the related *MacTag* lengths and allowed numbers of failed MAC verifications for each value of the HMAC key.

A commonly acceptable length for the *MacTag* is 64 bits; *MacTags* with lengths shorter than 64 bits are discouraged.

5.4 Hash-based Key Derivation Functions

Cryptographic hash functions can be used as building blocks in key derivation functions (KDFs) (e.g., as specified in SP 800-56A, SP 800-56B, SP 800-56C and SP 800-108).

The KDFs specified in SPs 800-56A and B use the hash functions either directly, or indirectly in an HMAC construction. The method specified in SP 800-56C is an additional approved method for key derivation purposes in SPs 800-56A and B. The key derivation functions in SP 800-56A, B and C are used to generate (i.e., derive) secret keying material from a shared secret computed during a key agreement transaction between communicating parties.

The hash-based KDFs specified in SP 800-108 use an HMAC construction and can be used either: (1) to derive keying material from an existing key, or (2) as a key-expansion step in the key derivation method specified in SP 800-56C.

In addition to the KDFs in SPs 800-56A, B and C, and in SP 800-108, there are several other allowed application-specific KDFs described in SP 800-135 [SP 800-135]. These application-specific KDFs are approved for use in their own protocols with specific conditions; see SP 800-135 for detailed information.

5.4.1 Using a Hash Function Directly for Key Derivation

This section discusses and provides security requirements only for the KDFs in SP 800-56A and B that use the hash function directly as their building block (i.e., in the concatenation and ASN.1 KDFs). The KDFs derive the keying material from a shared secret computed during the key agreement transaction and other input attributes.

The security strength that can be provided by a derived key depends on the security strength of the asymmetric keys used to generate the shared secret, the preimage strength of the hash function used in the KDF and the length of the derived key. Therefore, if a derived key is intended to provide *s* bits of security strength, then each of the following **shall** be equal to or greater than *s*:

- The security strength provided by the asymmetric keys,
- The preimage strength of the hash function, and
- The length of the derived key in bits.

5.4.2 Using HMAC for Key Derivation During a Key Agreement Transaction

This section discusses and provides security requirements for the key derivation methods in SP 800-56A, B and C that use the hash function in an HMAC construction.

The KDFs specified in SPs 800-56A and B derive keys in a single step, and are being revised to allow the use of HMAC with an **approved** hash function for key derivation. SP 800-56C specifies a two-step key derivation procedure in which HMAC can be used during the key derivation process.

5.4.2.1 Using HMAC in the Single-Step Key Derivation Process

SP 800-56A and B specify a concatenation and ASN.1 form of a single-step key derivation function.

The security strength that can be provided by a derived key depends on the security strength of the asymmetric keys used to generate the shared secret, the preimage strength of the hash function used in the HMAC construction used in the KDF and the length of the derived key. Note that in this case, the key used for HMAC is a salt, which can be a publicly-known value, a secret value, or a combination of both. Therefore, if a derived key is intended to provide *s* bits of security strength, then each of the following **shall** be equal to or greater than *s*:

- The security strength provided by the asymmetric keys,
- The preimage strength of the hash function used in the HMAC construction, and
- The length of the derived key in bits.

5.4.2.2 Using HMAC in the Two-Step Key Derivation Process

SP 800-56C specifies a two-step key derivation procedure, which is also known as an "extraction-then-expansion" procedure. This procedure is included by reference in SPs 800-56A and B.

The extraction-then-expansion procedure in SP 800-56C is comprised of two separate steps: randomness-extraction and key-expansion, both of which can be implemented using an HMAC construction; when this construction is used, the same hash function is used for the randomness extraction and key expansion steps.

The security strength that can be provided by a derived key depends on the security strength of the asymmetric keys used to generate the shared secret, the preimage strength of the hash function used in each HMAC construction and the length of the derived key. In the randomness extraction step, the key used for HMAC is a salt, which can be a publicly-known value, a secret value, or a combination of both. In the key expansion step, the key is the entire output of the randomness extraction step, e.g., if SHA-1 is used during the randomness extraction step, then the output is 160 bits in length, and is used as the key derivation key for the key expansion step.

When this two-step procedure is used, if a derived key is intended to provide *s* bits of security strength, then each of the following **shall** be equal to or greater than *s*:

- The security strength provided by the asymmetric keys,
- The preimage strength of the hash function used in the HMAC construction of each step of the procedure, and
- The length of the derived key in bits.

5.4.3 Using HMAC for Key Derivation from a Pre-shared Key

This section discusses and provides security requirements for the KDFs in SP 800-108 that are used to derive keying material from a pre-shared (i.e., existing) key, called a key

derivation key. HMAC can be used with an **approved** hash function to derive keying material.

The security strength of a derived key depends on the security strength provided by the key derivation key, the hash function used in the HMAC construction and the length the the derived key. If a derived key is intended to provide *s* bits of security strength, then each of the following **shall** be equal to or greater than *s*:

- The security strength provided by the key derivation key,
- The preimage strength of the hash function used in the HMAC construction, and
- The length of the derived key in bits.

5.5 Random Number (Bit) Generation

A random bit generator (RBG) is used to produce random bits. These bits may be used directly or may be converted to a random number (integer). Approved RBGs that use deterministic algorithms[4], along with methods for converting a bit string to an integer can be found in SP 800-90A. Other approved random number generators were either specified or approved by reference in FIPS 186-2 [FIPS 186-2]; however, their use is limited (see SP 800-131A [SP 800-131A]).

RBGs may be constructed using cryptographic hash functions. The hash function used by the RBG **shall** be selected so that the RBG can provide a security strength that meets or exceeds the minimum security strength required for the random bits that it generates. See SP 800-57_Part 1 for the security strength that can be provided for each approved hash function for random number generation.

6 References

[SP 800-38B] NIST Special Publication (SP) 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, May 2005.

[SP 800-56A] NIST Special Publication (SP) 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, March 2007.

[SP 800-56B] NIST Special Publication (SP) 800-56B, Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography, August 2009.

[SP 800-56C] NIST Special Publication (SP) 800-56C, Recommendation for Key Derivation through Extraction-then-Expansion, (Draft) published July 2011.

[4] Commonly known as deterministic random bit generators or pseudorandom number generators.

[SP 800-57] NIST Special Publication (SP) 800-57, Part 1, Recommendation for Key Management: General, (Draft) published May 2011.

[SP 800-90A] NIST Special Publication (SP) 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, (Draft revision) May 2011.

[SP 800-106] NIST Special Publication (SP) 800-106, Randomized Hashing for Digital Signatures, February 2009.

[SP 800-108] NIST Special Publication (SP) 800-108, Recommendation for Key Derivation Using Pseudorandom Functions, November 2008.

[SP 800-131A] E. Barker and A. Roginsky, "Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths", NIST Special Publication 800-131A, January 2011.

[SP 800-135] NIST Special Publication (SP) 800-135, Recommendation for Existing Application-Specific Key Derivation Functions, December 2010.

[FIPS 180-4] Federal Information Processing Standard 180-4, Secure Hash Standard (SHS), (Draft) February 2011.

[FIPS 186-2] Federal Information Processing Standard 186-2, Digital Signature Standard (DSS), January 2000.

[FIPS 186-3] Federal Information Processing Standard 186-3, Digital Signature Standard (DSS), June 2009.

[FIPS 198-1] Federal Information Processing Standard 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*, July 2008.

[SHA1 Attack] Wang X., Yin Y., and Yu H., *Finding Collisions in the Full SHA-1*, The 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 2005.

[Second Preimage Attack] Kelsey J. and Schneier B., *Second Preimages on n-bit hash functions for Much Less than 2^n Work*, Lecture Notes in Computer Science, Vol. 3494, Springer, 2005, ISBN-10 3-540-25910-4.

[BCK1] M. Bellare, R. Canetti, and H. Krawczyk, *Keyed Hash Functions for Message Authentication*, Proceedings of Crypto'96, LNCS 1109, pp. 1-15. (http://www.research.ibm.com/security/keyed-md5.html)

Appendix A : Actual Second Preimage Resistance Strengths of Approved Cryptographic Hash Functions

In an application, if the size of the messages is small, then the second preimage resistance strengths of the cryptographic hash functions are practically the same as their preimage resistance strengths described Section 4.2.

The actual second preimage resistance strength for SHA-1, SHA -224, SHA-256 and SHA-512 is ($L - M$), where L is the output block size of the cryptographic hash function, and M is a function of the input block size, as follows:

$$M = log_2\left(\frac{max\_message\_length\_in\_bits}{input\_block\_size\_in\_bits}\right)$$

when $max\_message\_length\_in\_bits \geq input\_block\_size\_in\_bits$.

Max_message_length_in_bits is the permitted maximum length of input messages in bits, and *input_block_size_in_bits* is the input block size of the cryptographic hash function.

For example, if a message that is 2^{33} bits in length (i.e., a gigabyte long) is hashed by SHA-256 (whose input block size is 2^9 bits), the second preimage resistance is ($L - M$) = (256 – 24) = 232 bits (where $L = 256$, and $M = log_2(2^{33}/2^9) = (33 - 9) = 24$).

It is important to note that the amount of work is based on the number of message blocks processed, not on the number of entire messages processed.

The actual second preimage resistance strength of SHA-1, SHA -224, SHA-256 and SHA-512 varies, depending on the maximum size of the messages in the application using the cryptographic hash function.

The second preimage resistance of SHA-384, SHA-512/224 or SHA-512/256 does not depend on the message length because the attack in [Second Preimage Attack] requires work of more than 2^{384}, 2^{224} or 2^{256} respectively, but to break the second preimage resistance of SHA-384, SHA-512/224 or SHA-512/256, the brute force attack requires only the work of 2^{384}, 2^{224} or 2^{256} respectively.

Note: SHA-256 and SHA-224 are essentially the same cryptographic hash function. Each produces an "initial" output of 256 bits, but from different initial values. In the case of SHA-224, the "initial" output is truncated to 224 bits. Therefore, L is 256 bits for these two cryptographic hash functions when determining their actual second preimage resistance strengths.

For any truncated message digest of λ bits, the actual second preimage resistance strength of SHA-1, SHA-224, SHA-256, and SHA-512 is the minimum of ($L - M$) and λ, where $\lambda \leq 160$ for SHA-1, $\lambda \leq 224$ for SHA-224, $\lambda \leq 256$ for SHA-256, and $\lambda \leq 512$ for SHA-512.
.

The actual preimage resistance strengths of SHA-384, SHA-512/224 and SHA-512/256 is λ (where $\lambda \leq 384$ for SHA-384, $\lambda \leq 224$ for SHA-512/224, and $\lambda \leq 256$ for SHA-512/256).

Appendix B : Document Changes

The original of this document was published in February 2009. The main technical additions to this revision are:

1) Adding and addressing security properties of SHA-512/224 and SHA-512/256.
2) Expanding the discussion of the security of HMAC values in Section 5.3 and,
3) Section 5.4: Hash-based Key Derivation Function was rewritten to incorporate the "extraction-then-expansion" key derivation procedure specified in SP 800-56C and to discuss different approved hash-based key derivation functions.