NIST Special Publication 800-107

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

February 2009

U.S. Department of Commerce
Otto J. Wolff, Acting Secretary

National Institute of Standards and Technology
Patrick D. Gallagher, Deputy Director

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) 180-3. These include functions such as digital signature applications, Keyed-hash Message Authentication Codes (HMACs) and Hash-based Key Derivation Functions (HKDFs).

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

1 Introduction 2
2 Authority 2
3 Glossary of Terms, Acronyms and Mathematical Symbols 3
3.1 Terms and Definitions 3
3.2 Acronyms 6
3.3 Symbols 6
4 Approved Hash Algorithms 6
4.1 Hash Function Properties 6
4.2 Strengths of the Approved Hash Algorithms 8
5 Cryptographic Hash Function Usage 9
5.1 Truncated Message Digest 9
5.2 Digital Signatures 10
5.2.1 Full-length Message digests 11
5.2.2 Truncated Message digests 11
5.2.3 Randomized Hashing for Digital Signatures 11
5.3 Keyed-Hash Message Authentication Codes (HMAC) 12
5.3.1 Description 12
5.3.2 The HMAC Key 12
5.3.3 Truncation 13
5.3.4 Security of the HMAC Algorithm 13
5.3.5 Security of HMAC Values 13
5.4 Hash-based Key Derivation Functions (HKDFs) 14
5.5 Random Number (Bit) Generation 15
6 References 15
Appendix A 17
Appendix B 17

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) 180-3, the Secure Hash Standard (SHS) [FIPS 180-3], specifies five approved hash algorithms: SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512. 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 Federal Information Processing Standard (FIPS) 180-3 [FIPS 180-3], such as digital signature applications [FIPS 186-3], Keyed-hash Message Authentication Codes (HMACs) [FIPS 198-1] and Hash-based Key Derivation Functions (HKDFs) [SP 800-56A] & [SP 800-56B].

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 Module Validation Program (CMVP), a joint

effort of NIST and the Communications Security Establishment of the Government of Canada.

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 180-3].
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 180-3].
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 of a message. 4. The verification of a digital signature.
Message authentication code (MAC)	A fixed length bit string, computed by a MAC algorithm, that is used to establish the authenticity and, hence, the integrity of a message.
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
HKDF	Hash-based 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* or truncated message digest.
$\| x \|$	The length (in bits) of the bit string x. For example, $\|01100100\| = 8$.

4 Approved Hash Algorithms

Currently, there are five approved hash algorithms, SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512, which are specified in [FIPS 180-3]. These hash algorithms produce outputs of 160, 224, 256, 384 and 512 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, L, produced by a given cryptographic hash function. For example, SHA-256 produces a (full-length) hash value of 256 bits; SHA-256 provides an estimated collision resistance of 128 bits.

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

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, *L*, produced by a given cryptographic hash function. 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 the 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, *L*, produced by a given cryptographic hash function. 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 applications (as specified in [FIPS 198-1]). In the case of digital signatures, SHA-1 does not provide 112 bits of collision resistance needed to achieve the security strength. On the other hand, SHA-1 does provide 112 bits of preimage resistance that is needed to achieve

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

the security strength for HMAC. The security strengths of the approved cryptographic hash functions for different applications can be found in [SP 800-57].

4.2 Strengths of the Approved Hash Algorithms

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

	SHA-1	**SHA-224**	**SHA-256**	**SHA-384**	**SHA-512**
Collision Resistance Strength in bits	< 80[3]	112	128	192	256
Preimage Resistance Strength in bits	160	224	256	384	512
Second Preimage Resistance Strength in bits	105-160	201-224	201-256	384	394-512

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

As mentioned in Section 4.1, the estimated collision resistance strength of any approved cryptographic hash function is half the length of its hash value. Currently, SHA-224, SHA-256, SHA-384 and SHA-512 are believed to have collision resistance strengths of 112, 128, 192 and 256 bits (half of the lengths of their hash values), respectively. However, the latest cryptanalytic results for SHA-1 [SHA1 Attack] indicate that it may have a collision resistance strength that is considerably less then its expected strength of 80 bits.

The estimated preimage resistance strengths of SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 are 160, 224, 256, 384 and 512 bits (the lengths of the hash values),

[3] Current estimated value is around 60.

respectively. 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, 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 when 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 the Appendix A. In the case of SHA-384, the second preimage resistance strength does not depend on the message length; details can be found in the Appendix A.

Note that the preimage resistance and the second preimage resistance of any approved hash algorithm specified in [FIPS 180-3] are stronger than its collision resistance.

5 Cryptographic Hash Function Usage

5.1 Truncated Message Digest

Some applications may require a message digest that is shorter than the (full-length) message digest provided by an approved cryptographic hash function specified in [FIPS 180-3]. 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.

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

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 the SHA-256 cryptographic hash function

are selected as the truncated message digest, and the rightmost 160 bits of the message digest are discarded.

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 the 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. NIST-approved cryptographic hash functions are believed to provide the collision resistant strengths as specified in the Table 1 of Section 4.1.

For digital signature applications, the security strength of a hash function is normally its collision resistance strength. When appropriate processing is applied to the data before it is hashed, 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]. 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 then 80 bits (see Table 1 in Section 4.1). Therefore, SHA-1 **should not** be used in any new digital signature applications that require at least 80 bits of security. Furthermore, SHA-1 **shall not** be used in any digital signature applications after the end of 2010. 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].

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

5.2.1 Full-length Message digests

The cryptographic hash functions specified in [FIPS 180-3] (i.e., SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512) generate (full-length) message digests of 160, 224, 256, 384 and 512 bits, respectively. Full-length message digests as specified in [FIPS 180-3] can be used with approved digital signature algorithms as specified in [FIPS 186-3]. The estimated strength of collision resistance provided by a hash-function is half the length of the full-length message digest.

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. SHA-224, SHA-256, SHA-384 and SHA-512 could be used to generate a 224-bit message digest, although, in the case of SHA-224, the hash-value would not be truncated.

5.2.3 Randomized Hashing for Digital Signatures

As described in Section 5.2, the security strength of a digital signature application [FIPS 186-3] is limited to the collision resistance strength of the cryptographic hash function. However, when using the randomized hashing technique specified in [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 80 bits of security strength.

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; 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 80 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) (message receiver(s)) that need to verify the transmitted *MacTag*.

The HMAC output is generated from the 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 by the HMAC generator, generates a *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 on the security strength of the HMAC key, *K*. An HMAC key shall be generated such that its security strength 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 "authentic" *MacT*ags (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 [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

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 8 bits. For example, a low bandwidth channel or a desired high efficiency computation application such as audio or video casting application might use 8-bit *MacTags*.

5.3.4 Security of the HMAC Algorithm

The security strength of the HMAC algorithm[4] is the minimum of the security strength of K and the value of $2L$ (i.e., *security strength* = min(*security strength of K*, $2L$)). For example, if the security strength of K is 128 bits, and SHA-1 is used, then the security strength of the HMAC algorithm is 128 bits.

The HMAC key K **shall** be generated with a security strength that meets or exceeds the desired security strength of the HMAC application, and the approved hash algorithm in the HMAC application **shall** have a message digest length of at least half of the desired security strength (in bits) of the HMAC application. For example, if the desired security strength of the HMAC application is 256 bits, the HMAC key K **shall** be generated with a security strength of at least 256 bits, and an approved hash function with the message digest length of at least 256/2 (128) bits **shall** be used.

5.3.5 Security of 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

[4] As described in [BCK1], success of a collision attack on any approved HMAC algorithm [FIPS 198-1] 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.

key), the security strength provided by a *MacTag* depends on its length. The length of a *MacTag* **shall** be sufficiently long to prevent false acceptance of forged data. For most applications, a length of 64 to 96 bits is sufficient.

Shorter MacTags may also be acceptable if the rate of false acceptances does not create a significant impact for the application. For example, in video/audio stream applications, accepting one bad data package in 2^8 data packages may not create a huge impact for the application.

5.4 Hash-based Key Derivation Functions (HKDFs)

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] and [SP 800-108]). KDFs using cryptographic hash functions as their building blocks are called Hash-based Key Derivation Functions (HKDFs). The main purpose of an HKDF is to generate (i.e., derive) secret keys from a secret value (e.g., a shared key, or a shared secret in a key agreement scheme) that is shared between communicating parties. The security strengths of the derived secret keys are limited to the security strength of the secret value. The security strength of the secret value **shall** meet or exceed the desired security strengths of the derived secret keys.

Each of the two approved HKDFs in [SP 800-56A] and [SP 800-56B] uses a secret value (i.e., a shared secret) that is shared between two communicating parties, an approved cryptographic hash function and other input attributes to derive secret keying material, such as HMAC keys.

There are two approved uses for these HKDFs:

1) The HKDF is used to generate secret keys from a shared secret that is generated from one of the approved key agreement methods specified in [SP 800-56A] and [SP 800-56B]. In this case, the key agreement method **shall** support a security strength equal to or greater than the desired security strength of the secret keys to be derived using the HKDF.
2) The HKDF is used to generate secret keys from an existing shared key. In this case, the security strength of the existing shared key **shall** meet or exceed the desired security strength of the secret keys to be derived. An existing shared key may be one of the derived secret keys described in (1) above.

In addition to the two HKDFs in [SP 800-56A] and [SP 800-56B], there are other approved KDFs specified in NIST Special Publication 800-108, *Recommendation for Key Derivation Using Pseudorandom Functions* [SP 800-108]. In [SP 800-108], one of the approved KDFs uses an HMAC algorithm as a Pseudorandom Function (PRF). The Key Derivation Key specified in [SP 800-108] is used as the HMAC key to the HMAC function. The Key Derivation Key may be one of the generated secret keys as described in (1) and (2) above, or an existing shared key. The Key Derivation Key **shall** meet or exceed the desired security strength of the secret keys to be derived.

Additional allowed KDFs are listed in [IMPGUIDE].

The security strengths provided by approved hash functions when used for KDFs are specified in [SP 800-57]. The security strength of an approved hash function used by the HKDF **shall** be equal to or greater than the highest required security strength for the generated keying material.

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[5], along with methods for converting a bit string to an integer can be found in [SP 800-90].

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 maximum security strength required for the random bits that it generates. See [SP 800-57] 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, (Draft) December 2008.

[SP 800-57] NIST Special Publication (SP) 800-57, Part 1, Recommendation for Key Management: General, August 2005.

[SP 800-90] NIST Special Publication (SP) 800-90, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, June 06.

[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.

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

[FIPS 180-3] Federal Information Processing Standard 180-3, Secure Hash Standard (SHS), October 2008.

[FIPS 186-3] Federal Information Processing Standard 186-3, Digital Signature Standard (DSS), (Draft) November 2008.

[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.

[IMPGUIDE] Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program, available at http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf

[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 above.

The actual second preimage resistance strength of any of the approved hash functions, except SHA-384, 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 $M = log_2(2^{33}/2^9) = (33 - 9) = 24$).

The actual second preimage resistance strength of any approved cryptographic hash function, except SHA-384, varies, depending on the maximum size of the messages in the application using the cryptographic hash function.

The second preimage resistance of SHA-384 does not depend on the message length because the attack in [Second Preimage Attack] requires the work of more than 2^{384}, and to break the second preimage resistance of SHA-384, the required work is only 2^{384}.

Note: SHA-256 and SHA-224 are the same cryptographic hash function. Each produces an output of 256 bits, but from different initial values. In the case of SHA-224, the 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. The actual preimage resistance strength of SHA-384 is λ ($\lambda \leq 384$).

Appendix B

The changes in this document from its previous version are: the 4th footnote was added in Section 5.3.4 (Security of the HMAC Algorithm) and the reference information of [BCK1] in the footnote was added at the end of References section.