NIST Special Publication 800-56C

Recommendation for Key Derivation through Extraction-then-Expansion

Lily Chen

Computer Security Division
Information Technology Laboratory

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

November 2011

U.S. Department of Commerce
John E. Bryson, Secretary

National Institute of Standards and Technology
Patrick D. Gallagher, Under Secretary for Standards and Technology and Director

Abstract

This Recommendation specifies techniques for the derivation of keying material from a shared secret established during a key establishment scheme defined in NIST Special Publications 800-56A or 800-56B through an extraction-then-expansion procedure.

KEY WORDS: key derivation, extraction, expansion

Acknowledgements

The author, Lily Chen of the National Institute of Standards and Technology (NIST), would like to acknowledge the authors, Yevgeniy Dodis, Rosario Gennros, Johan Håstad, Hugo Krawczyk, and Tal Rabin, of Crypto 2004 paper titled "Randomness extraction and key derivation using CBC, cascade and HMAC modes [12]" for formalizing the idea of extraction-then-expansion key derivation. Especially, the author would like to acknowledge Hugo Krawczyk for introducing the instantiation of extraction-then-expansion with HMAC as presented in [10] and [11].

The author like to thank her colleagues, Elaine Barker, Quynh Dang, Sharon Keller, John Kelsey, Allen Roginsky, Meltem Sonmez Turan, and Tim Polk of NIST, Miles Smid of Orion Security Solutions, and Rich Davis of the National Security Agency, for helpful discussions and valuable comments.

The author gratefully appreciates the thoughtful and instructive comments received during the public comment periods, which helped to improve the quality of this publication.

Authority

This document 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 guideline 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 nongovernmental 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 these guidelines 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.

Table of Contents

1. Introduction 6
2. Scope and Purpose 6
3. Definitions, Symbols and Abbreviations 6
   3.1 Definitions 6
   3.2 Symbols and Abbreviations 8
4. Outline of Extraction-then-Expansion Key Derivation 9
5. Randomness Extraction 9
6. Key Expansion 13
7. Summary and Discussion 14
Appendix A: References 16
Appendix B: Conformance to "Non-testable" Requirements 17

Figures

Figure 1: Extraction-then-Expansion Procedure 9

1. Introduction

During an execution of some of the public-key-based key establishment schemes specified in NIST Special Publications 800-56A [1] and 800-56B [2], a key derivation method is used to obtain secret cryptographic keying material. This Recommendation specifies an alternative key derivation method to be used in a key establishment scheme specified in 800-56A and 800-56B.

2. Scope and Purpose

This Recommendation specifies a two-step key derivation procedure, as one of the **approved** key derivation methods, that employs an extraction-then-expansion technique for deriving keying material from a shared secret generated during a key establishment scheme specified in [1] or [2]. Several application-specific key derivation functions that use **approved** variants of this extraction-then-expansion procedure are described in NIST Special Publication 800-135 [5].

The key derivation procedure specified in this Recommendation consists of two steps: 1) randomness extraction (to obtain a single key derivation key) and 2) key expansion (to derive keying material with a desired length from the key derivation key). Since NIST Special Publication 800-108 [4] specifies several families of key derivation functions that are **approved** for deriving additional keying material from a given cryptographic key derivation key, those functions are employed in the second (key expansion) step of the procedure.

3. Definitions, Symbols and Abbreviations

3.1 Definitions

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 or 3) specified in a list of NIST-approved security functions.
Hash function	A function that maps a bit string of arbitrary length to a fixed-length bit string. **Approved** hash functions are designed to satisfy the following properties: 1. (One-way) It is computationally infeasible to find any input that maps to any pre-specified output, and 2. (Collision resistant) It is computationally infeasible to find any two distinct inputs that map to the same output. **Approved** hash functions are specified in FIPS 180 [9].

Key derivation	A process that derives keying material from a key or a shared secret.
Key derivation key	A key that is used as input to the key expansion step to derive other keys. In this Recommendation, the key derivation key is obtained by performing randomness extraction on a shared secret.
Key establishment	A procedure that results in generating shared keying material among different parties.
Key expansion	The second step in the key derivation procedure specified in this Recommendation to derive keying material with the desired length.
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.
Message authentication code (MAC)	A family of cryptographic algorithms that is parameterized by a symmetric key. Each of the algorithms can act on input data (called a message) of an arbitrary length to produce an output value of a specified length (called the *MAC* of the input data). A MAC algorithm can be used to provide data origin authentication and data integrity protection. In this Recommendation, a MAC algorithm is also called a MAC function.
Nonce	A time-varying value that has at most a negligible chance of repeating – for example, a random value that is generated anew for each use, a timestamp, a sequence number, or some combination of these.
Pseudorandom function	A function that can be used to generate output from a secret random seed and a data variable, such that the output is computationally indistinguishable from truly random output. In this Recommendation, an **approved** message authentication code (MAC) is used as a pseudorandom function in the key expansion step, where a key derivation key is used as the secret random seed.
Randomness extraction	The first step in the key derivation procedure specified in this Recommendation, which produces a key derivation key from a shared secret.
Salt	A byte string that is used as an input in the randomness extraction step specified in Section 5.
Shared secret	A value generated during a public-key-based key establishment scheme defined in NIST SP 800-56A [1] or SP 800-56B [2].
Shall	A requirement that needs to be fulfilled to claim conformance to this Recommendation. Note that **shall** may be coupled with **not** to become **shall not**.

Should	An important recommendation. Ignoring the recommendation could result in undesirable results. Note that **should** may be coupled with **not** to become **should not**.

3.2 Symbols and Abbreviations

AES	Advanced Encryption Standard (as specified in FIPS 197 [7]).
AES-CMAC(k, M)	An untruncated AES-CMAC computed over message *M* using key *k* with AES block cipher.
CMAC	Cipher-based Message Authentication Code (as specified in NIST SP 800-38B [8]).
ECC	Elliptic curve cryptography.
FFC	Finite field cryptography.
h	An integer whose value is the length of the output of the PRF in bits.
hash( )	A hash function.
HMAC	Keyed-hash Message Authentication Code (as specified in FIPS 198-1 [6]).
HMAC-*hash*(*k*, *M*)	An untruncated HMAC computed over message *M* using key *k* with hash function *hash* [9].
IFC	Integer factorization cryptography
K_{DK}	A key derivation key that is used as an input in the key expansion step specified in Section 6.
K_M	Keying material that is derived from a key derivation key K_{DK} and other data through the key expansion step.
L	An integer specifying the length of the derived keying material K_M in bits, which is represented as a binary string when it is an input to a key derivation procedure.
$[L]_2$	The binary representation of an integer *L* with a specific binary length, which is specified through the encoding method for the input data to a key derivation procedure.
MAC	Message Authentication Code.
PRF	Pseudorandom Function.
s	Salt used during randomness extraction.
Z	Shared secret.

0x00	An all-zero octet. In this Recommendation, it is suggested for use as an ending indicator of a variable length data field, which holds the ASCII code for a character string.

4. Outline of Extraction-then-Expansion Key Derivation

The extraction-then-expansion key derivation procedure specified in this Recommendation begins with a shared secret Z that is a byte string established during an execution of a public-key-based key establishment scheme specified in NIST SP 800-56A [1] or NIST SP 800-56B [2].

The randomness extraction step uses HMAC as defined in FIPS 198-1 [6] or AES-CMAC as defined in NIST SP 800-38B [8], with a byte string s (called the *salt*) as the "key", and the shared secret Z as the "message". The output of the randomness extraction step is a key derivation key K_{DK}.

The key expansion step uses the key derivation key K_{DK} and other information exchanged and/or pre-shared, such as identifiers for the involved parties, protocol identifiers, and the labels of the derived keys, as the input to an **approved** key derivation function specified in SP 800-108 [4] to produce secret keying material K_M of a desired length L. The extraction-then-expansion procedure is shown in Figure 1.

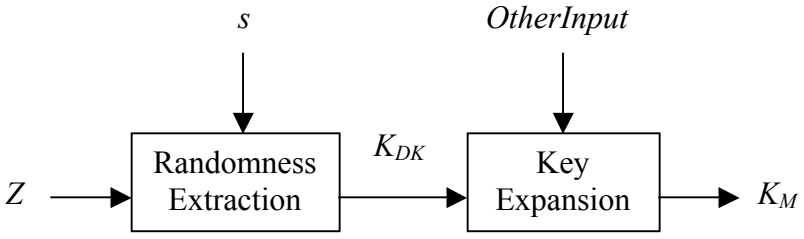

Figure 1: Extraction-then-Expansion Procedure

In a key establishment scheme defined in [1] or [2], the above key derivation procedure is called with input s, Z, and *OtherInput*.

IETF RFC 5869 [10] describes an instantiation of the above extraction-then-expansion key derivation procedure using HMAC for the extraction and expansion steps. For extensive rationale on the key derivation through the extraction-then-expansion procedure specified in this Recommendation see [11].

5. Randomness Extraction

The key establishment schemes specified in 800-56A [1] and 800-56B [2] require the selection of a parameter set with a targeted security strength for key establishment (see SP 800-57, Part 1 [3] for a discussion on security strengths). These parameter sets are defined for finite field cryptography (FFC), elliptic curve cryptography (ECC) and integer factorization cryptography (IFC). During the execution of some key establishment

schemes, a shared secret is generated and subsequently used as input to a key derivation method. The two-step method specified herein begins with the randomness extraction step, in which a MAC function (either HMAC or AES-CMAC) is used to obtain a key derivation key from the shared secret. The output length (in bits) of the MAC function used for randomness extraction **shall** be at least as large as the targeted security strength (in bits) of the parameter set employed by the key establishment scheme (see Tables 1-3 below).

When used for message authentication, the output of an HMAC or AES-CMAC function may be truncated to a specific length that is smaller than the output length of the hash function *hash* (in case of HMAC-*hash*) or the block length of AES (in case of AES-CMAC). In this Recommendation, an untruncated output of the MAC function in the extraction step is used as K_{DK}. Tables 1, 2, and 3 identify the MAC functions that can be used in the extraction step when a particular parameter set is employed by the key establishment scheme. In these tables, "✓" indicates that the output length of the untruncated MAC function can support the targeted security strength and can be used in the extraction step, and "×" indicates that the output length of the untruncated MAC function is too short to support the targeted security strength and **shall not** be used in the extraction step.

Table 1 - FFC parameter sets and appropriate MAC functions for randomness extraction

FFC Parameter Set Name		FA	FB	FC
Targeted security strength (in bits)		80	112	112
Binary length of field order *p*		1024	2048	2048
Binary length of subgroup order *q*		160	224	256
MAC Functions	HMAC-SHA-1	✓	✓	✓
	HMAC-SHA-224 & HMAC-SHA-512/224	✓	✓	✓
	HMAC-SHA-256 & HMAC-SHA-512/256	✓	✓	✓
	HMAC-SHA-384	✓	✓	✓
	HMAC-SHA-512	✓	✓	✓
	AES-CMAC	✓	✓	✓

Table 2 - ECC parameter sets and appropriate MAC functions for randomness extraction

ECC Parameter Set Name		EA	EB	EC	ED	EE
Targeted security strength (in bits)		80	112	128	192	256
Binary length of ECC subgroup order *n*		163-223	224-255	256-383	384-511	512+
	HMAC-SHA-1	✓	✓	✓	×	×
	HMAC-SHA-224 & HMAC-SHA-512/224	✓	✓	✓	✓	×

MAC Functions	HMAC-SHA-256 & HMAC-SHA-512/256	✓	✓	✓	✓	✓
	HMAC-SHA-384	✓	✓	✓	✓	✓
	HMAC-SHA-512	✓	✓	✓	✓	✓
	AES-CMAC	✓	✓	✓	×	×

Table 3 - IFC Parameter sets and appropriate MAC functions for randomness extraction

IFC Parameter Set Name		IA	IB
Targeted security strength (in bits)		80	112
Binary length of modulus *n*		1024	2048
MAC Functions	HMAC-SHA-1	✓	✓
	HMAC-SHA-224 & HAMC-SHA-512/224	✓	✓
	HMAC-SHA-256 & HMAC-SHA-512/256	✓	✓
	HMAC-SHA-384	✓	✓
	HMAC-SHA-512	✓	✓
	AES-CMAC	✓	✓

When an untruncated HMAC-*hash* algorithm is employed in the randomness extraction step, the output length of HMAC-*hash* for any **approved** hash function *hash* meets the security strength requirements for use with each of the FFC and IFC parameter sets, as shown in Tables 1 and 3, respectively. As indicated in Table 2, however, for a key establishment scheme using elliptic curve cryptography, HMAC-SHA-1 (which produces an output of only 160 bits) **shall not** be used with parameter sets ED and EE, which are intended to support security strengths up to 192 bits and 256 bits, respectively.

A CMAC-based randomness extraction step uses AES-CMAC, an instantiation of the CMAC function employing the AES block cipher. AES can use 128, 192, and 256-bit keys. However, the untruncated output length of AES-CMAC is 128 bits, since this is the AES block size; therefore, the security strength that can be supported by an AES-CMAC-based randomness extraction step is limited to 128 bits. AES-CMAC with all the specified key sizes is acceptable for use in the randomness extraction step for a key establishment scheme with all parameter sets using FFC or IFC, as presented in Tables 1 and 3, respectively. However, as shown in Table 2, an **approved** key establishment scheme using elliptic curve cryptography with parameter set ED or EE **shall not** use AES-CMAC for randomness extraction, because of the restriction on the security strength that can be supported by the CMAC output length.

In the extraction step, the following notations are used.

- *MAC* – The MAC function used during the extraction step, chosen in conformance with Tables 1, 2, and 3. MAC is either HMAC-*hash*, an (untruncated) instantiation of the HMAC function employing the hash function *hash*, or AES-CMAC, an (untruncated) instantiation of the CMAC function employing the AES block cipher.
- *s* – Salt, a (public or secret) byte string used as the "key" during the execution of the randomness extraction step. The length of *s* is determined by the MAC function used in the extraction step as shown below.
    - HMAC-*hash* algorithms as defined in [6] can accommodate keys of any length up to the maximum bit length permitted for input to the hash function *hash*. As specified in [6], if the bit length of an HMAC key is greater than the input block length for *hash*, that key is replaced by its hash value.
    - AES-CMAC requires key lengths to be 128, 192, or 256 bits. The bit length of the salt *s* **shall** be the same length as the AES key (i.e., 128, 192, or 256 bits).

  The salt could be, for example, a value computed from nonces exchanged as part of a key establishment protocol, a value already shared by the protocol participants, or a value that is pre-determined by the protocol. If there is no means to select a salt and to share it with all participants, then the salt **shall** be the all-zero byte string. If HMAC-*hash* is used, the bit length of the all-zero byte string **shall** equal that of the input block for *hash*. If AES-CMAC is used, the bit length of the all-zero byte string **shall** equal the length of the AES key used. For further discussion of the salt, see Section 7.
- *Z* – A shared secret established during an execution of an **approved** public key-based key establishment scheme. It is represented as a byte string and used as the "message" in a MAC execution in the randomness extraction step. Each call to the randomness extraction step requires a freshly-computed shared secret *Z*, and this shared secret **shall** be zeroized immediately following its use in the extraction process.
- K_{DK} – The output of the randomness extraction step. When HMAC-*hash* is used, it is a binary string of length *h*, where *h* is the output length in bits of the hash function *hash*. When AES-CMAC is used, it is a binary string of length 128 bits.

Once a MAC function has been selected, the extraction step is performed as follows.

Input: *s* and *Z*.

Process:

1. $K_{DK} = MAC(s, Z)$, where *MAC* is the selected HMAC-*hash* or AES-CMAC function.
2. Zeroize *Z*.

Output: K_{DK}.

K_{DK} is used as the key derivation key in the key expansion step discussed in Section 6.

6. Key Expansion

Key expansion is the second step in the key derivation procedure specified in this Recommendation. This step employs the key derivation key K_{DK}, which is obtained through the randomness extraction step specified in Section 5, to produce keying material K_M of the desired length L.

One of the PRF-based key derivation functions defined in NIST SP 800-108 [4] **shall** be used in the key expansion step. These key derivation functions employ a MAC function (either HMAC or AES-CMAC) as the PRF. In this Recommendation, the PRF used in key expansion step is determined by the MAC function used in the extraction step. Specifically,

- if an HMAC-*hash* is used in the randomness extraction step, then the same HMAC-*hash* (with the same hash function *hash*) is used as the PRF in the key expansion step; and
- if an AES-CMAC (with key length 128, 192, or 256 bits) is used in the randomness extraction step, then AES-CMAC with a 128-bit key is used as the PRF in the key expansion step.

The rationale for these rules is discussed in Section 7.

The key derivation key K_{DK} is used as K_I in the selected KDF mode from SP 800-108. In other words, K_{DK} is used as the key in HMAC-*hash* or AES-CMAC for the expansion step. Components of the "messages" (some fixed, some variable) that are input during the iterated execution of HMAC-*hash* or AES-CMAC are determined by the protocol that uses the key derivation procedure specified in this Recommendation. Input components that are fixed during the iterated PRF evaluations of the expansion step may include the following data fields.

1. *Label* – A binary string that identifies the purpose for the derived keying material. For example, it can be the ASCII code for a character string. The value and encoding method used for the *Label* are defined in a larger context, for example, in the protocol that uses this key derivation procedure.
2. *Context* – A binary string containing the information related to the derived keying material. When a static key pair can be used in more than one key establishment scheme, the Context **should** include a scheme identifier that is unique to the scheme employed during the particular key establishment transaction that invoked the key expansion process. If the information is available, *Context* **should** include the identifiers of the parties who are deriving and/or using the derived keying material and, optionally, a nonce known by the parties who derive the keys. *Context* is equivalent to the data field "*OtherInfo*" used by the single-step key derivation functions defined in NIST SP 800-56A and SP 800-56B. (See Section 5.8 of SP 800-56A or Section 5.9 of SP 800-56B for suggested formats for "*OtherInfo*".)

3. L – An integer specifying the length (in bits) of the derived keying material K_M. L is represented as a binary string $[L]_2$ when it is used to form input to the key expansion step. The length of the binary string $[L]_2$ is specified by the encoding method for the input data. (L is equivalent to "*keydatalen"* in the single-step key derivation functions defined in NIST SP 800-56A [1] and to "*Kbits*" in the single-step key derivation functions defined in NIST SP 800-56B [2].)

For the inputs to the key expansion step, each data field **shall** be encoded unambiguously. When concatenating the above encoded data fields, the length for each data field and the order for the fields may be defined as a part of a key expansion specification or by the protocol where the key expansion step is used.

When using one of the KDF modes defined in SP 800-108 for the key expansion step, the fixed portion of the message input during execution of HMAC-*hash* or AES-CMAC could, for example, be represented as $P = Label \mid\mid 0x00 \mid\mid Context \mid\mid [L]_2$ (i.e., a concatenation of a *Label*, which is assumed to be the ASCII code for a character string; an ending indicator of *Label*, *0x00*; *Context*; and $[L]_2$). Other formats are allowed, as long as they are well-defined by the key expansion implementation or by the protocol employing this key derivation procedure.

Depending on the specific SP 800-108 KDF mode, the messages input to HMAC-*hash* or AES-CMAC may also include a counter (in counter mode), a key block derived during the previous execution in the same pipeline or another pipeline of HMAC-*hash* or AES-CMAC (in feedback mode and double pipeline iteration mode), or both a counter and a key block.

The derived secret keying material K_M **shall** be computed in its entirety before outputting any portion of it. The key derivation key K_{DK} **shall** be zeroized immediately upon the completion of the key derivation procedure.

7. Summary and Discussion

In this section, the following issues are discussed:

1. Pairing MAC functions for extraction and expansion

This Recommendation approves both HMAC-*hash* with an **approved** hash function *hash* and AES-CMAC with AES-128, AES-192, or AES-256 as randomness extraction algorithms.

When an HMAC-*hash* is used for the extraction step, the same HMAC-*hash* (i.e., with the same hash function *hash*) is specified for use as the PRF in the key-expansion step, even though technically, an HMAC function implemented with any **approved** hash function would be capable of accepting the untruncated output of the HMAC-*hash* used in the extraction step as its key. Similarly, when HMAC-*hash* is used for the extraction step AES-CMAC may not be used for expansion, even though a CMAC function implemented with AES-256 would be capable of accepting the untruncated output of an HMAC-*hash* when implemented with SHA-256, and other constructs could be devised by truncating the output of HMAC-hash.

If AES-CMAC is used in the extraction step, using any AES key length (i.e., 128, 192 or 256 bits), the output key-derivation key K_{DK} is 128 bits long. In this case, AES-CMAC with a 128-bit key is specified for use as the PRF in the key expansion step, even though it would be technically possible to employ an HMAC-*hash* as the PRF in key expansion after AES-CMAC was used for the extraction step. On the other hand, because a salt is used as a key in the randomness extraction step and the AES-CMAC output is 128 bits long, using AES with 192-bit and 256-bit key will not increase the security strength for the randomness extraction. As a result, when AES-CMAC is used, using AES-CMAC with 128 bit key in both extraction and expansion steps is a more appropriate choice.

These restrictions are designed to promote interoperability, reduce complexity, and simplify in-line negotiation of key derivation functions without any reduction in security.

2. Using a truncated hash function for HMAC

SHA-224, SHA-512/224, SHA-512/256 and SHA-384 are **approved** hash functions specified in [9]. SHA-224 is a truncated version of SHA-256, while SHA-512/224, SHA-512/256, and SHA-384 are truncated versions of SHA-512. Each of these truncated versions uses a specific initial value, which is different from the untruncated version. HMAC with a truncated hash function may not be a practical choice for use in extraction or expansion, owing to the additional truncation steps that are performed during execution. This may be particularly counterproductive when the expansion step outputs multiple blocks of keying material. In this case, instead of using the truncated output for each block of keying material, it is more efficient to use the untruncated output of HMAC-SHA-256 or HMAC-SHA-512.

As a general rule, protocol developers are encouraged to make use of the untruncated output of HMAC-SHA-256 or HMAC-SHA-512.Since cryptographic modules supporting the truncated output can support the untruncated output with little additional expense, this recommendation is expected to maximize interoperability and performance.

3. The salt used in the extraction step

As defined in this Recommendation, each randomness extraction step uses a salt value as an input (see Section 5.1 and 5.2). If there is no means to select a salt and to share it among the key establishment participants, then an all-zero byte string **shall** be specified as the default salt value. Using the all-zero byte string is equivalent to not using a salt in the extraction process, as discussed in Section 3.1 of IETF RFC 5869[10], “To salt or not to salt.”

Appendix A: References

[1] NIST SP 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, March 2008.

[2] NIST SP 800-56B, Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography, August 2009.

[3] NIST SP 800-57, Recommendation for Key Management Part1: General, May 2011.

[4] NIST SP 800-108, Recommendation for Key Derivation using Pseudorandom Functions, October 2009.

[5] Draft NIST SP 800-135, Recommendation for Existing Application-Specific Key Derivation Functions, December 2010.

[6] FIPS 198-1, The Keyed-Hash Message Authentication Code (HMAC), July 2008.

[7] FIPS 197, Advanced Encryption Standard, November 2001.

[8] NIST SP 800-38B, Recommendation for Block Cipher Modes of Operation – The CMAC Mode for Authentication, May 2005.

[9] Draft FIPS 180-4, Secure Hash Standard, 2011.

[10] IETF RFC 5869 HMAC-based Extract-and-Expand Key Derivation Function (HKDF), May 2010.

[11] H. Krawczyk. "Cryptographic Extraction and Key Derivation: The HKDF Scheme", Advances in Cryptology - Crypto'2010, Lecture Notes in Computer Science Vol. 6223, pp. 631-648. Springer. 2010.

[12] Y. Dodis, R. Gennros, J. Håstad, H. Krawczyk, and T. Rabin. "Randomness Extraction and Key Derivation Using the CBC, Cascade, and HMAC modes". Advances in Cryptology - Crypto'2004, Lecture Notes in Computer Science Vol. 3152, pp. 494-510. Springer. 2004.

Appendix B: Conformance to "Non-testable" Requirements

This appendix lists those requirements whose conformance is not testable.

Table 1 - List of "non-testable" requirements

Section	Requirement
Authority	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.
5	If there is no means to select a salt and to share it with all participants, then the salt **shall** be the all-zero byte string If HMAC-*hash* is used, the bit length of the all-zero byte string **shall** equal that of the input block for *hash*. If AES-CMAC is used, the bit length of the all-zero byte string **shall** equal the length of the AES key used.
5	Each call to the randomness extraction step requires a freshly computed shared secret Z, and this shared secret **shall** be zeroized immediately following its use in the extraction process.
6	For the inputs to the key expansion step, each data field **shall** be encoded unambiguously.
6	The derived secret keying material K_M **shall** be computed in its entirety before outputting any portion of it.
6	The key derivation key K_{DK} **shall** be zeroized immediately upon the completion of the key derivation procedure.
7	If there is no means to select a salt and to share it among the key establishment participants, then an all-zero byte string **shall** be specified as the default salt value.