DRAFT 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

September 2010

U.S. Department of Commerce
Gary Locke, Secretary, Secretary

National Institute of Standards and Technology
Patrick D. Gallagher, 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 and 800-56B through an extraction-then-expansion procedure.

KEY WORDS: key derivation, extraction, expansion

Acknowledgements

The author, Lily Chen of National Institute of Standards and Technology (NIST), would like to thank her colleagues, Elaine Barker, Quynh Dang, and Tim Polk of NIST, and Rich Davis of the National Security Agency, for helpful discussions and valuable comments.

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.

Conference testing for implementations of this Recommendation will be conducted within the framework of the Cryptographic Module Validation Program (CMVP) and the Cryptographic Algorithm Validation Program (CAVP), which are joint effort of NIST and the Communication Security Establishment of the Government of Canada. The requirements of this Recommendation are indicated by the word “shall”. Some of these requirements may be out-of-scope for CMVP and CAVP testing, and thus are the responsibility of entities using, implementing or installing applications that incorporate this Recommendation.

Table of Contents

Figures

1. Introduction

A key derivation method **shall** be applied to a shared secret obtained during an execution of a public-key-based key-establishment scheme as specified in NIST Special Publication 800-56A and 800-56B ([1] and [2]), in order to obtain cryptographic keys. This Recommendation specifies key derivation procedures needed to transform the shared secret into keying material with the property that each non-overlapping segment with a required length can be used as a cryptographic key.

2. Scope and Purpose

This Recommendation specifies a two-step key-derivation procedure that employs an extraction-then-expansion technique as described in [9]. This procedure is **approved** for deriving keying material from a shared secret. Several application-specific key derivation functions that use **approved** variants of this extraction-then-expansion procedure are described in NIST Special Publication 800-135 [4].

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 ([3]) specifies several families of key derivation functions that are **approved** for deriving additional keying material from a given cryptographic key, those functions are employed in the second (key-expansion) step of the procedure.

The key-establishment schemes specified in [1] and [2] incorporate the use of specific key derivation functions that combine the extraction of randomness from a shared secret and the generation of keying material into a one-step process. The key derivation functions defined in [1] and [2] are also **approved** for deriving keying material from a shared secret.

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 180-3 [8].
Key agreement	A key establishment procedure where the resultant secret keying material is a function of information contributed by two participants, so that no party can predetermine the value of the secret keying material independently from the contributions of the other party. Contrast with key transport.
Key derivation	The process that derives keying material from a key or a shred secret value.
Key derivation key	A key used as an input to a key derivation function to derive other keys. In this Recommendation, a key derivation key is obtained by applying a randomness extraction function to a shared secret.
Key establishment	The procedure that results in shared secret keying material among different parties.
Key expansion	The second step in the key derivation procedure specified in this Recommendation to derive keying material.
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 initiate initialization vectors.
Key transport	A procedure, conducted by two or more participants, whereby one party (the sender) selects a secret value for the keying material and then securely distributes that value to other parties, after which the resultant keying material is shared by all participants.
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 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.
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 random seed and a data variable, such that the output is computationally indistinguishable from truly random output.
Randomness extraction	The first step in the key derivation procedure specified in this Recommendation that derives a key derivation key from a shared secret.
Salt	A byte string that is used as an input in the randomness extraction step.
Shared secret	A value generated during a public-key-based key-establishment scheme defined in NIST SP 800-56A or SP 800-56B.
Shall	This term is used to indicate 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	This term is used to indicate 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	Advance Encryption Standard (as specified in FIPS 197 [6]).
AES-CMAC (k, M)	A CMAC of message *M* using key *k* with AES block cipher.
CMAC	Cipher-based Message Authentication Code (as specified in NIST SP 800-38B [7]).
DLC	Discrete logarithm cryptography
h	An integer whose value is the length of the output of the PRF in bits.
hash( )	A hash function.
HMAC-*hash*(*k*, *M*)	Keyed-hash Message Authentication Code (as specified in FIPS 198-1 [5]) of message *M* using key *k* with hash function *hash*.
K_{DK}	A key-derivation key that is used as an input in the key expansion step specified in Section 6. For key derivation, K_{DK} is the key-derivation key used (along with other data) to derive keying material K_M.
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 function.

MAC	Message Authentication Code.
PRF	Pseudorandom Function.
s	Salt used during randomness extraction.
Z	Shared secret

4. Outline of Extraction-then-Expansion Key Derivation

An extraction-then-expansion key-derivation procedure begins with a shared secret *Z* that has been established during an execution of a public-key-based key-establishment scheme, such as those specified in [1] and [2]. The key-derivation procedure requires that the shared secret Z be a byte string. When implementing key agreement in conformance with SP 800-56A, the shared secret output from a DLC primitive employed by a key-establishment scheme is converted from its original form (an integer representation of an element of a finite field or some representation of a point on an elliptic curve) to a byte string, using one of the conversion routines specified in that document.

The randomness-extraction step uses HMAC as defined in FIPS 198-1 [5] or AES-CMAC as defined in NIST SP 800-38B [7], 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 pre-shared and exchanged information by both parties, such as identifiers for the involved parties, protocol identifiers, and labels of the derived keys, as the input to an **approved** key derivation function to produce keying material K_M of a desired length *L*.

The extraction-then-expansion procedure is shown in Figure 1.

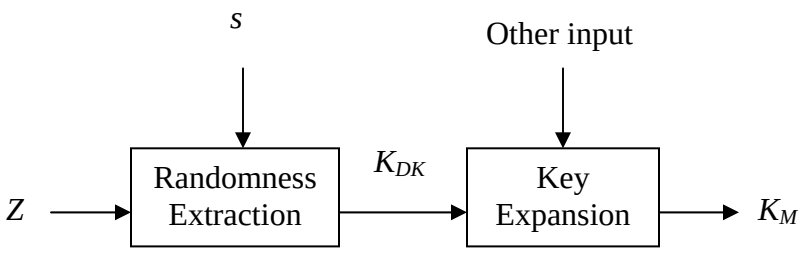

Figure 1: Extraction-then-Expansion Procedure

.

5. Randomness Extraction

This section defines an **approved** randomness-extraction step using HMAC or AES-CMAC.

5.1 HMAC-based Randomness Extraction

An HMAC-based randomness extraction procedure uses HMAC-*hash*, an instantiation of the HMAC function employing the hash function *hash*. The output length of the selected hash function *hash* **shall** meet the requirements specified in Section 5.5.1 of SP 800-56A. The following notations are used.

1) *s* – Salt, a (public or private) byte string used as the key for HMAC-hash during the execution of the randomness-extraction step. 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 are no other means to select the salt, then it shall be an all-zero byte string whose bit length equals that of the input blocks for *hash*.
2) *Z* – A shared secret established during an execution of a public-key-based key-establishment scheme. It is represented as a byte string and used as a message in an HMAC-*hash* execution in the randomness extraction step.
3) K_{DK} – The output of the randomness extraction step. It is a binary string of length *h*, where *h* is the output length in bits of the hash function *hash* used in HMAC-*hash*.

The randomness-extraction step is described as follows.

Input: *s* and *Z*.

Process:

1. K_{DK} = HMAC –*hash* (*s, Z*)

Output: K_{DK}.

K_{DK} will be used as a key-derivation key in the key-expansion step discussed in Section 6.

5.2 CMAC-based Randomness Extraction

A CMAC-based randomness extraction procedure uses AES-CMAC, an instantiation of the CMAC function employing the AES block cipher. AES can use 128, 192, and 256-bit keys. The output length of CMAC is 128 bits. The following notations are used.

1) *s* – Salt, a (public or private) byte string used as the key for AES-CMAC during the execution of the randomness-extraction step. Its bit length must be 128, 192, or 256 -- matching the key length selected for the AES block cipher. The salt could be, for example, a value computed from nonce 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 are no other means to select the salt, then it shall be an all-zero byte string.
2) *Z* – A shared secret established during an execution of a public-key-based key-establishment scheme. It is represented as a byte string and used as a message in an AES-CMAC execution in the randomness extraction step.

3) K_{DK} – The output of the randomness extraction step. It is a binary string of length 128 bits. The randomness-extraction step is described as follows.

Input: *s* and *Z*.

Process:

1. $K_{DK} = AES\text{-}CMAC\ (s, Z)$

Output: K_{DK}.

K_{DK} will be used as a key-derivation key in the key-expansion step introduced in the next section.

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} obtained through the randomness extraction step specified in Section 5 to produce keying material K_M of a desired length *L*.

One of the key-derivation functions defined in NIST SP 800-108 [3] **shall** be used in the key-expansion step. If an HMAC-*hash* is used in the randomness extraction, then the HMAC-*hash* with the same hash function *hash* **shall** be used as the PRF in key expansion. If AES-CMAC is used in the randomness extraction, then AES-CMAC with a 128 bit key **shall** be used as the PRF in key expansion. The rationale for this requirement is discussed in Section 7.

Since the key expansion step is to be executed in a key establishment scheme as specified in SP 800-56A ([1]) and SP 800-56B ([2]), the following discussion of the input to the key-derivation function uses the terminologies used in [1] and [2].

K_{DK} is used as the key in HMAC-*hash*[1] or AES-CMAC for the expansion step. The message inputs to each execution of HMAC-*hash* (some fixed, some variable) is determined by the protocol that uses the key-derivation procedure specified in this Recommendation and by the selected iteration mode. Fixed input for each iteration may include the following data fields:

1. *Label* – A binary string that identifies the purpose for the derived keying material. The value and encoding method used for the *Label* is 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 used during the key agreement scheme can be used in other key agreement schemes, the *Context* **shall** include an identifier for the scheme being used for this key agreement transaction. If the information is available, it **should** include the identities 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.

[1] In NIST SP 800-108, K_{DK} is denoted as K_I.

(*Context* **should** be equivalent to the data field "*OtherInfo*" used by the key derivation functions defined in NIST SP 800-56A and SP 800-56B.)

3. *L* – An integer specifying the length (in bits) of the derived keying material K_M. *L* is represented as a binary string when it is used to form input to the key expansion. The length of the binary string is specified by the encoding method for the input data. (*L* is equivalent to "*keydatalen"* in the key derivation functions defined in NIST SP 800-56A ([1]) and to "*Kbits*" in the key derivation functions defined in NIST SP 800-56B ([2]).) See Section 5.8 of SP 800-56A for a suggested format to form "*OtherInfo*".

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 an order for the fields may be defined as a part of a key expansion specification or by the protocol where the key expansion is used. Between the variable length data fields, an all zero octet *0x00* may be used as an indicator.

When using one of the iteration modes defined in SP 800-108 for key expansion, the fixed input message in each execution of HMAC-*hash* or AES-CMAC can be represented as $P = Label \,||\, 0x00 \,||\, Context \,||\, [L]_2$ (i.e., a concatenation of a *Label*, a separation indicator, *0x00*, the *Context*, and $[L]_2$). The order for these fields used in this paragraph is one example. Other orderings 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 mode of iteration, the input to HMAC-hash or AES-CMAC may also include a counter (in counter mode), a key block derived in the previous execution of HMAC-*hash* or AES-CMAC (in feedback mode), or both.

7. Discussion

This Recommendation approves HMAC-*hash* with an **approved** hash function *hash* and AES-CMAC with AES-128, 192, and 256 as randomness extraction algorithms. A PRF selected for key expansion **shall** use all the output bits of the randomness extraction as input to the expansion step.

When an HMAC-*hash* is used for the extraction step, any **approved** hash function could be used for the HMAC-hash of the expansion step. However, for practical reasons, the same hash function is specified in this Recommendation for both steps.

If AES-CMAC is used in the extraction step, then the output key-derivation key K_{DK} is 128 bits long. In this case, AES-CMAC with 128-bit key **shall** be used for the key-expansion step. While it is technically possible to employ an HMAC-*hash* as the PRF for key-expansion after AES was used for the extraction step, for practical reasons, this Recommendation specifies that AES be used for both steps.

Appendix A: References (Informative)

[1] NIST SP 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, May 2006.

[2] NIST SP 800-56B, Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography, expected to be published in 2008.

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

[4] Draft NIST SP 800-135 "Recommendation for Application Specific Key Derivations." To appear 2010.

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

[6] FIPS 197, Advanced Encryption Standard, 2001.

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

[8] FIPS 180-3, Secure Hash Standard, 2008.

[9] 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.