Draft NIST Special Publication 800-108
Revision 1

Recommendation for Key Derivation Using Pseudorandom Functions

Lily Chen

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-108r1-draft

National Institute of
Standards and Technology
U.S. Department of Commerce

Draft NIST Special Publication 800-108 Revision 1

Recommendation for Key Derivation Using Pseudorandom Functions

Lily Chen
Computer Security Division
Information Technology Laboratory

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-108r1-draft

October 2021

U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
James K. Olthoff, Performing the Non-Exclusive Functions and Duties of the Under Secretary of Commerce for Standards and Technology & Director, National Institute of Standards and Technology

Authority

This publication has been developed by NIST in accordance with its statutory responsibilities under the Federal Information Security Modernization Act (FISMA) of 2014, 44 U.S.C. § 3551 *et seq.*, Public Law (P.L.) 113-283. NIST is responsible for developing information security standards and guidelines, including minimum requirements for federal information systems, but such standards and guidelines shall not apply to national security systems without the express approval of appropriate federal officials exercising policy authority over such systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130.

Nothing in this publication should be taken to contradict the 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. This publication may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright in the United States. Attribution would, however, be appreciated by NIST.

National Institute of Standards and Technology Special Publication 800-108 Revision 1
Natl. Inst. Stand. Technol. Spec. Publ. 800-108 Rev. 1, 26 pages (October 2021)
CODEN: NSPUE2

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-108r1-draft

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at https://csrc.nist.gov/publications.

Public comment period: *October 18, 2021* through *January 18, 2022*

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
Email: sp800-108-comments@nist.gov

All comments are subject to release under the Freedom of Information Act (FOIA).

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in federal information systems. The Special Publication 800-series reports on ITL's research, guidelines, and outreach efforts in information system security, and its collaborative activities with industry, government, and academic organizations.

Abstract

This Recommendation specifies techniques for the derivation of additional keying material from a secret key—either established through a key establishment scheme or shared through some other manner—using pseudorandom functions HMAC, CMAC, and KMAC.

Keywords

CMAC; HMAC; key derivation; KMAC; pseudorandom function.

Acknowledgments

The author, Lily Chen of the National Institute of Standards and Technology (NIST) would like to thank her colleagues – Elaine Barker and Meltem Sönmez Turan of NIST and Rich Davis of the National Security Agency – for their helpful discussions and valuable comments. The author also gratefully appreciates the comments received during the development of the previous version of this publication.

Call for Patent Claims

This public review includes a call for information on essential patent claims (claims whose use would be required for compliance with the guidance or requirements in this Information Technology Laboratory (ITL) draft publication). Such guidance and/or requirements may be directly stated in this ITL Publication or by reference to another publication. This call also includes disclosure, where known, of the existence of pending U.S. or foreign patent applications relating to this ITL draft publication and of any relevant unexpired U.S. or foreign patents.

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either:

a) assurance in the form of a general disclaimer to the effect that such party does not hold and does not currently intend holding any essential patent claim(s); or

b) assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft publication either:

    i. under reasonable terms and conditions that are demonstrably free of any unfair discrimination; or
    ii. without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interest.

The assurance shall also indicate that it is intended to be binding on successors-in-interest regardless of whether such provisions are included in the relevant transfer documents.

Such statements should be addressed to: sp800-108-comments@nist.gov

Table of Contents

List of Appendices

List of Figures

1 Introduction

When a party obtains a cryptographic key, additional keys will often be needed. There are numerous methods for obtaining the keying material required by **approved** cryptographic algorithms (see SP 800-133 Rev. 2 [1] for a discussion of the recommended techniques). The requisite keying material is often obtained from the output of a key-derivation function that takes a preexisting cryptographic key (and other data) as input. Key-derivation functions are used to derive additional keys from a cryptographic key.

The key derivation functions specified in the original edition (2008) of NIST Special Publication (SP) 800-108[1] used pseudorandom functions HMAC and CMAC. In Revision 1, KDF using KMAC is added in Section 5.4.

2 Scope and Purpose

This Recommendation specifies several families of key derivation functions that use pseudorandom functions. These key derivation functions can be used to derive additional keys from an existing cryptographic key that was previously established through an automated key-establishment scheme (e.g., as defined in SP 800-56A [2] and SP 800-56B [3]), previously generated (e.g., using a pseudorandom bit generator as specified in SP 800-90A [4] or a previous instance of key derivation as specified in this Recommendation), and/or previously shared in some other way (e.g., by manual distribution).

Effectively, the key derivation functions specified in this Recommendation provide the key expansion functionality described in [5], where key derivation is portrayed as a process that potentially requires two separate steps: 1) randomness extraction (to obtain an initial key) and 2) key expansion (to produce additional keys from that initial key and other data).

[1] Chen L (2008) Recommendation for Key Derivation Using Pseudorandom Functions. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-108.

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, 2) adopted in a FIPS or NIST Recommendation, or 3) specified in a list of NIST-approved security functions.
cryptographic key	A bit string used as a secret parameter by a cryptographic algorithm. In this Recommendation, a cryptographic key is either a truly random bit string of a length specified by the cryptographic algorithm or a pseudorandom bit string of the specified length that is computationally indistinguishable from one selected uniformly at random from the set of all bit strings of that length.
entity	An individual (person), organization, device, or a combination thereof. In this Recommendation, an entity may be a functional unit that executes certain processes.
hash function	A function that maps a bit string of arbitrary length to a fixed-length bit string. Approved hash functions satisfy the following properties: 1. (Collision resistance) It is computationally infeasible to find any two distinct inputs that map to the same output. 2. (Preimage resistance) Given a randomly chosen target output, it is computationally infeasible to find any input that maps to that output. (This property is called the one-way property.) 3. (Second preimage resistance) Given one input value, it is computationally infeasible to find a second (distinct) input value that maps to the same output as the first value. This Recommendation uses the strength of the preimage resistance of a hash function as a contributing factor when determining the security strength provided by a key-derivation method.
key derivation	The process by which keying material is derived from 1) either a cryptographic key or a shared secret produced during a key-agreement scheme and 2) other data. This Recommendation specifies key derivation from a cryptographic key.
key-derivation function	A function that, with the input of a cryptographic key and other data, generates a bit string called the keying material, as defined in this Recommendation.

key-derivation key (KDK)	A key used as an input to a key derivation function to derive additional keying material.
key establishment	A procedure conducted by two or more participants, after which the resultant keying material is shared by all participants.
key hierarchy	A multiple-level tree structure such that each node represents a key, and each branch – pointing from one node to another – indicates a key derivation from one key to another key.
keying material	A bit string such that non-overlapping segments of the string (with the required lengths) can be used as cryptographic keys or other secret (pseudorandom) parameters.
message authentication code (MAC)	A family of secret-key cryptographic algorithms acting on input data of arbitrary length to produce an output value of a specified length (called the MAC of the input data), which can be employed to provide authentication of the origin of data and/or data-integrity protection. In this Recommendation, approved MAC algorithms are used to determine families of pseudorandom functions (indexed by the choice of key) that are employed during key derivation.
mode of iteration	A method for iterating the multiple invocations of a pseudorandom function in order to derive the keying material with a required length.
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.
pipeline	A term used to describe a series of sequential executions of a pseudorandom function.
pseudorandom function	An indexed family of (efficiently computable) functions, each defined for the same particular pair of input and output spaces. (For the purposes of this Recommendation, one may assume that both the index set and the output space are finite.) The indexed functions are pseudorandom in that if a function from the family is selected by choosing an index value uniformly at random, and one's knowledge of the selected function is limited to the output values corresponding to a feasible number of (adaptively) chosen input values, then the selected function is computationally indistinguishable from a function whose outputs were fixed uniformly at random.
security strength	A number characterizing the amount of work that is expected to suffice to "break" the security definition of a given cryptographic algorithm.

shall	The term used to indicate a requirement of a Federal Information Processing Standard (FIPS) or a requirement that needs to be fulfilled to claim conformance with this Recommendation. Note that **shall** may be coupled with **not** to become **shall not**.
should	The term 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

$A(i)$	The output of the i^{th} iteration in the first pipeline in a double pipeline iteration mode.
$A \| B$	The concatenation of bit strings A and B.
CMAC	Cipher-based Message Authentication Code (as specified in NIST SP 800-38B [6]).
h	The length of the PRF output in bits.
HMAC	Keyed-hash Message Authentication Code (as specified in FIPS 198-1 [7]).
i	The counter incremented following each iteration of PRF evaluation; it is represented as a bit string of length r when it is used as an input to the PRF.
IV	A bit string that is used as an initial value in computing the first iteration of the PRF in feedback mode. It may be an empty string.
KDF	Key Derivation Function.
$K(i)$	The output of the i^{th} iteration of the PRF.
K_I	A key-derivation key. K_I is used as input to a key-derivation function (along with other data) in order to derive the output keying material K_O.
K_O	Output keying material that is derived from the key-derivation key K_I and other data that were used as input to a key-derivation function.
KDK	Key-derivation key.

KMAC	Keccak-based Message Authentication Code (as specified in SP 800-185 [8]).
L	An integer specifying the length of the derived keying material K_O in bits, which is represented as a bit string when it is an input to a key-derivation function.
MAC	Message Authentication Code.
n	The number of iterations of the PRF needed to generate L bits of keying material.
PRF	Pseudorandom Function.
$PRF(s, x)$	A pseudorandom function with seed s and input data x.
r	An integer that is less than or equal to 32 and whose value is the length of the binary representation of the counter i when i is an input in counter mode or (optionally) in feedback mode and double-pipeline iteration mode of each iteration of the PRF.
$\|X\|$	The length of a bit string X in bits.
$[T]_2$	The length of an integer T when represented as a bit string.
w	The length of a key-derivation key in bits.
$\{X\}$	Used to indicate that data X is an optional input to the key-derivation function.
$\lceil X \rceil$	The smallest integer that is larger than or equal to X. The ceiling of X. For example, $\lceil 8.2 \rceil = 9$.
$X := Y$	X is defined to be equal to Y.
$\varnothing$	The empty bit string. That is, for any bit string A, $\varnothing \| A = A \| \varnothing = A$.
0x00	An all-zero octet.
$\in$	For an element s and a set S, $s \in S$, means s belongs to S.

4 Pseudorandom Function (PRF)

A pseudorandom function (PRF) is the basic building block in constructing a key derivation function in this Recommendation. Generally, a PRF family $\{PRF(s, x) \mid s \in S\}$ consists of polynomial time computable functions with an index (also called a seed) s and input x, such that when s is randomly selected from S and not known to observers, $PRF(s, x)$ is computationally indistinguishable from a random function defined on the same domain with output to the same range as $PRF(s, x)$. For a formal definition of a pseudorandom function, refer to [9].

When a cryptographic key K_I is regarded as the seed, that is, $s = K_I$, the output of the pseudorandom function can be used as keying material. In Section 5, several families of PRF-based key derivation functions are defined without describing the internal structure of the PRF. For key derivation, this Recommendation approves the use of the keyed-Hash Message Authentication Code (HMAC) specified in [7], the Cipher-based Message Authentication Code (CMAC) specified in [6], and the Keccak-based Message Authentication Code (KMAC) specified in [8] as the pseudorandom function. For a given KDF using HMAC, CMAC, or KMAC, the key K_I is assumed to be computationally indistinguishable from one that has been selected uniformly at random from the set of all of the bit strings with length of $|K_I|$.

Note that [5] specifies key-derivation methods that can be employed only as components of key-agreement schemes, as described in [2] and [3].

5 Key Derivation Function (KDF)

This section defines several families of key-derivation functions (KDF) that use PRFs. For the purposes of this Recommendation, a KDF is a function that – given input consisting of a (secret) key and other data – is used to generate (i.e., derive) keying material that can be employed by cryptographic algorithms. In other words, the KDFs specified here provide a key-expansion capability (as noted in Section 2).

The key that is input to a key-derivation function is called a key-derivation key (KDK). To comply with this Recommendation, a KDK **shall** be a cryptographic key (see Section 3.1). The KDK used as an input to one of the key derivation functions specified in this Recommendation can, for example, be generated by an approved cryptographic random bit generator (e.g., by a deterministic random bit generator of the type specified in [4] or output by an approved automated key-establishment scheme (e.g., as defined in [2] and [3]). The KDK can be a portion of the keying material derived from another KDK.

Note that the key-derivation methods employed as components of key-agreement schemes (as described in [2], [3], and [5]) include two-step methods in which the first step consists of extracting a KDK from a shared secret precursor. These extracted KDKs are not part of the output of a key-agreement scheme; they are only used to derive output keying material during a single execution of a scheme and then destroyed (along with all other sensitive, locally stored data associated with that particular execution).

In keeping with the usual terminology, the output of a key-derivation function is called the derived keying material and may subsequently be segmented into multiple keys. Any disjointed segments of the derived keying material (with the required lengths) can be used as cryptographic keys for the intended algorithms. However, in order to make sure that different parties will obtain the same keys from the derived keying material or, in the case where a single party derives the keying material, that re-derivation will generate the same keys (when required), the cryptographic application employing a KDF must define the way to convert (i.e., parse) the keying material into different keys. For example, when 256 bits of keying material are derived, the application may specify that the first 128 bits will be used as a key for a message authentication code and that the second 128 bits will be used as an encryption key for a given encryption algorithm.

Depending on the intended length of the keying material to be derived, the KDF may require multiple invocations of the PRF used in its construction. A method for iterating the multiple invocations is called a mode of iteration. In this Recommendation, a counter mode, a feedback mode, and a double pipeline mode are specified in Sections 5.1, 5.2, and 5.3, respectively.

In addition to these iteration modes, this Recommendation specifies a KDF using KMAC in Section 5.4. KMAC can output keying material that has the required length without iteration.

To define key-derivation functions, the following notations are used. Some of the notations have been defined in Section 3.2 and are repeated here for easy reference.

1) K_I – Key-derivation key; a key that is used as an input to a key derivation function (along with other input data) to derive keying material. When HMAC is used as the PRF, K_I is used as the HMAC key, and the other input data is used as the value of *text*, as defined in [7]. When CMAC is used as the PRF, K_I is used as the block cipher key, and the other input data is used as the message *M*, as defined in [6]. When KMAC is used as the PRF, K_I is used as the KMAC key, and the other input data is used as the main input string *X*, as defined in [8].

2) K_O – Keying material that is output from a key-derivation function specified in this Recommendation; a bit string of the required length that is derived using a key-derivation key (and other data).

3) *Label* – A string that identifies the purpose for the derived keying material, which is encoded as a bit string. The encoding method for the *Label* is defined in a larger context, for example, in the protocol that uses a KDF.

4) *Context* – A bit string containing the information related to the derived keying material. It may 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.

5) *IV* – A bit string that is used as an initial value in computing the first iteration in the feedback mode. It can be either public or secret. It may be an empty string. The length for an IV should be specified by the application or protocol using the key-derivation function.

6) *L* – An integer specifying the requested length (in bits) of the derived keying material K_O. *L* is represented as a bit string when it is an input to a key-derivation function. The length of the bit string is specified by the encoding method for the input data.

7) *h* – An integer that indicates the length (in bits) of the output of the PRF.

8) *n* – An integer whose value is the number of iterations of the PRF needed to generate *L* bits of keying material.

9) *i* – A counter; a bit string of length *r* that is an input to each iteration of a PRF in the counter mode and (optionally) in the feedback and double-pipeline iteration modes.

10) *r* – An integer ($r \leq 32$) that indicates the length of the binary representation of the counter *i*.

11) $\{X\}$ – Used to indicate that the data *X* is an optional input to the key-derivation function.

12) 0x00 – An all-zero octet; an optional data field that is used to indicate a separation of different variable-length data fields.[2]

For a PRF with an output length of h bits, the key-derivation function iterates the PRF n times, concatenating the outputs until L bits of keying material are derived; this requires $n = \lceil L/h \rceil$. When using counter mode, n **shall not** be larger than 2^r-1, where $r \leq 32$ is the length of the binary representations of the counter values. This ensures that the counter values are distinct, which is necessary to prevent a PRF used in counter mode from generating the same output. For feedback mode and double-pipeline iteration mode, a repeat in the counter value (if a counter is used at all) will not be sufficient to cause the iterated PRF to repeat an output value. Nevertheless, for compliance with this Recommendation, n **shall not** be larger than $2^{32}-1$ when using feedback mode or double-pipeline iteration mode; $L = (2^{32}-1)h$ bits of keying material is more than enough for most applications. Regardless of the mode, a particular implementation of a KDF or an application that uses a KDF can impose a smaller bound on the maximum value of n (the number of PRF iterations) than those imposed here.

For each of the iterations of the PRF, the key-derivation key K_I is used as the key, and the input data consists of an iteration-dependent input data and a string of fixed input data. Depending on the mode of iteration, the iteration-dependent input data could be a counter, the output of the PRF from the previous iteration, a combination of both, or an output from the first pipeline iteration (in the case of double-pipeline iteration mode). In the following key-derivation functions, the fixed input data is a concatenation of a *Label*, a separation indicator *0x00*, the *Context*, and $[L]_2$. One or more of these fixed input data fields may be omitted unless required for certain purposes, as discussed in Section 7.5 and Section 7.6.

The length for each data field and their order **shall** be defined unambiguously. For example, the length and the order may be defined as part of a KDF specification or by the protocol where the KDF is used. In each of the following sections, a specific order for the feedback value, the counter, the *Label*, the separation indicator *0x00*, the *Context*, and $[L]_2$ is used, assuming that each of them is represented with a specific length. This Recommendation specifies several families of KDFs. Alternative orders for the input data fields may be used for different KDFs.

5.1 KDF in Counter Mode

This section specifies a family of KDFs that uses the counter mode. In counter mode, the output of the PRF is computed with a counter as the iteration-dependent input data. The mode is defined as follows.

Parameters:

- h – The length of the output of the PRF in bits

[2] This indicator may be considered as a part of the encoding method for the input data and can be replaced by other indicators (e.g., an indicator to represent the length of the variable length field). If, for a specific KDF, only data fields with identical lengths are used, then the indicator may be omitted.

- r – The length of the binary representation of the counter i

Input: K_I, *Label*, *Context*, and L.

Process:

1. $n := \lceil L/h \rceil$.
2. If $n > 2^r-1$, then output an error indicator and stop (i.e., skip steps 3, 4, and 5).
3. $result := \emptyset$.
4. For i = 1 to n, do
   a. $K(i) := PRF(K_I, [i]_2 \,||\, Label \,||\, 0x00 \,||\, Context \,||\, [L]_2)$,
   b. $result = result \,||\, K(i)$.
5. $K_O :=$ the leftmost L bits of *result*.

Output: K_O.

In each iteration of PRF evaluation in step 4 above, the fixed input data is the string $Label \,||\, 0x00 \,||\, Context \,||\, [L]_2$. The counter $[i]_2$ is the iteration-dependent input data and is represented as a bit string of r bits. The KDF in counter mode is illustrated in Figure 1.

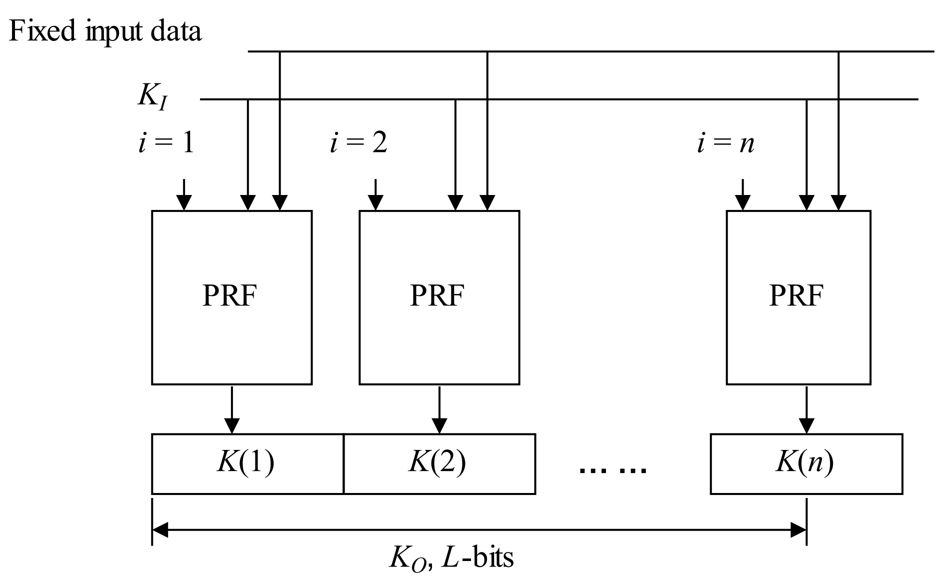

Figure 1. KDF in Counter Mode

5.2 KDF in Feedback Mode

This section specifies a family of KDFs that uses the feedback mode. In the feedback mode, the output of the PRF is computed using the result of the previous iteration and, optionally, using a counter as an iteration-dependent input data. The mode is defined as follows. (Note that when $L \le h$, $IV = \varnothing$, and the counter is used**,** the feedback mode will generate an output that is identical to the output of the counter mode specified in Section 5.1.)

Parameters:

- h – The length of the output of the PRF in bits
- r – The length of the binary representation of the counter i. r is specified only when a counter is used as an input

Input: K_I, *Label*, *Context*, *IV*, and L.

Process:

1. $n := \lceil L/h \rceil$.
2. If $n > 2^{32} - 1$, output an error indicator and stop (i.e., skip steps 3, 4, and 5).
3. $Result := \varnothing$ and $K(0) := IV$.
4. For $i = 1$ to n, do
   a. $K(i) :=$ PRF (K_I, $K(i\text{-}1)$ $\{|| [i]_2\}$ || *Label* || *0x00* || *Context* || $[L]_2$).
   b. $result := result \,||\, K(i)$.
5. $K_O :=$ the leftmost L bits of *result*.

Output: K_O.

In each iteration of PRF evaluation in step 4 above, the fixed input data is the string *Label* || *0x00* || *Context* || $[L]_2$. The iteration-dependent input data is $K(i\text{-}1)\{|| [i]_2\}$. The KDF in feedback mode is illustrated in Figure 2.

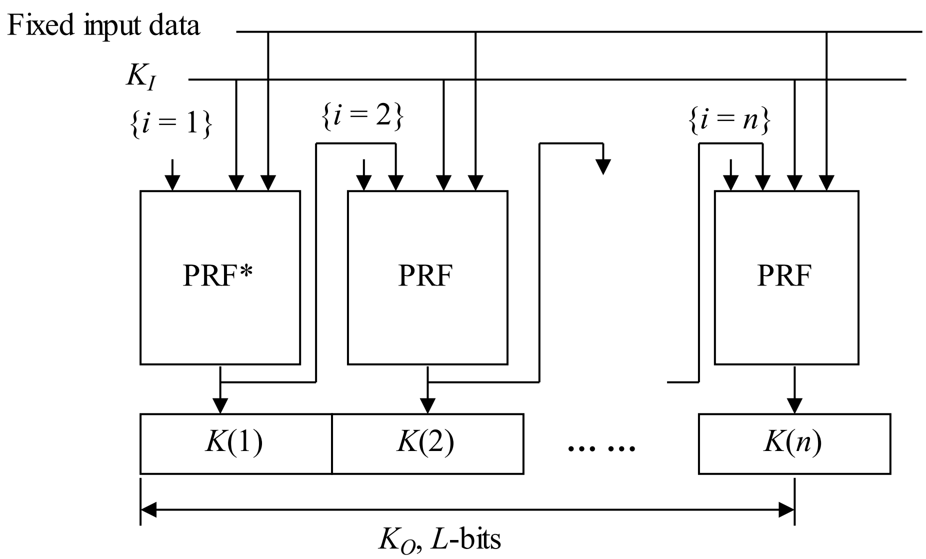

Figure 2. KDF in Feedback Mode

5.3 KDF in Double-Pipeline Mode

For a KDF in the counter mode or feedback mode, a PRF is iterated in a single pipeline. This section specifies a family of KDFs that iterates a PRF in two pipelines. In the first iteration pipeline, a sequence of secret values $A(i)$ is generated, each of which is used as an input to the respective PRF iteration in the second pipeline.

Parameters:

- h – The length of the output of the PRF in bits
- r – The length of the binary representation of the counter i. r is specified only when a counter is used as an input

Input: K_I, *Label*, *Context*, and L.

Process:

1. $n := \lceil L/h \rceil$.
2. If $n > 2^{32} - 1$, output an error indicator and stop (i.e., skip steps 3, 4, 5 and 6).
3. $Result := \emptyset$.
4. $A(0) := Label \,||\, 0x00 \,||\, Context \,||\, [L]_2$.
5. For $i = 1$ to n, do
   a. $A(i) :=$ PRF (K_I, $A(i\text{-}1)$).

b. $K(i) := \text{PRF}\ (K_I, A(i)\{\|\ [i]_2\}\|\ \mathit{Label}\ \|\ \mathit{0x00}\ \|\ \mathit{Context}\ \|\ [L]_2)$.

c. $\mathit{result} := \mathit{result}\ \|\ K(i)$.

6. K_O := the leftmost L bits of *result*.

Output: K_O.

The PRF iterations in the first pipeline use a feedback mode with input K_I and an initial value of $A(0) = \mathit{Label}\ \|\ \mathit{0x00}\ \|\ \mathit{Context}\ \|\ [L]_2$. Each PRF iteration in the second pipeline generates $K(i)$ from K_I and fixed input data while using $A(i)$ and, optionally, a counter $[i]_2$ as the iteration-dependent input data. The KDF in the double-pipeline iteration mode is illustrated in Figure 3.

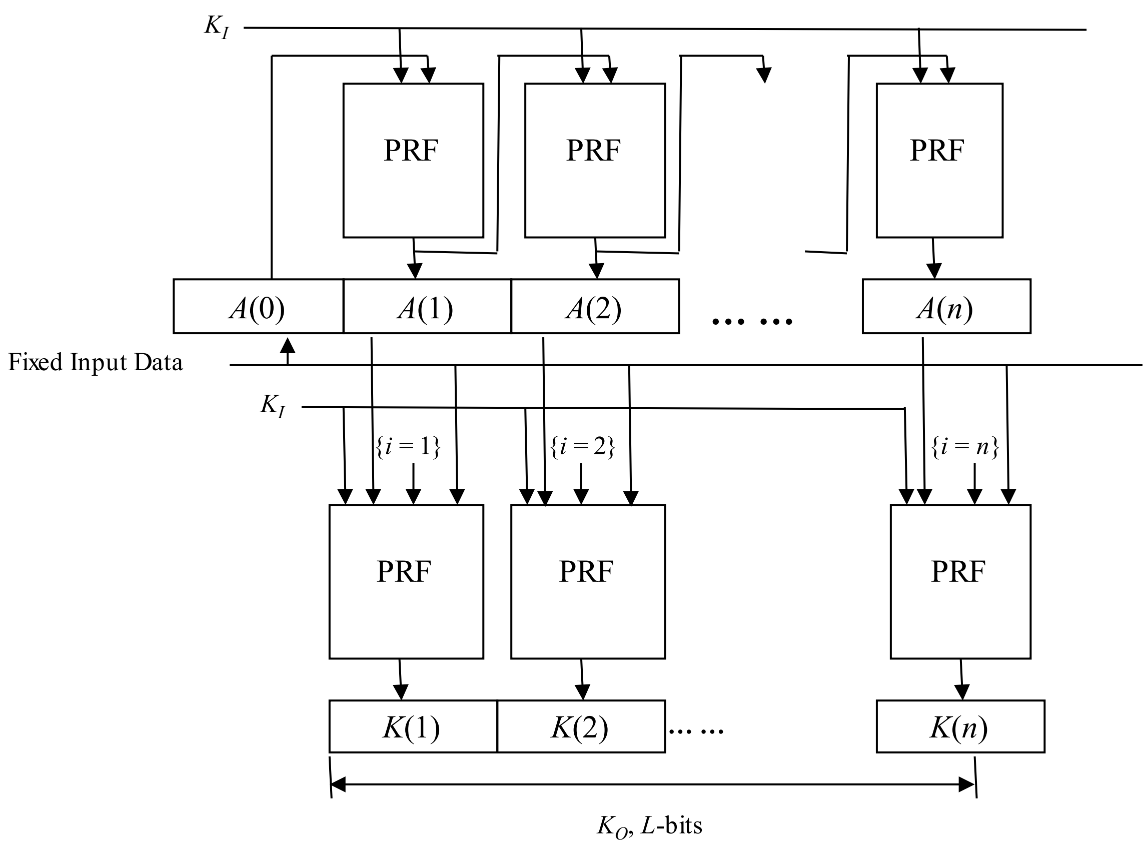

Figure 3. KDF in Double-pipeline Mode

5.4 KDF Using KMAC

KMAC is the Keccak-based Message Authentication Code, which is specified in [8]. KMAC is based on a sponge function and can output a bit string with a desired length L. When using KMAC, there is no need for iterated PRF evaluation (as was the case for the KDFs defined in Sections 5.1, 5.2, and 5.3). Two KMAC functions – KMAC128 and KMAC256 – are specified in [8]. Here, KMAC# means either KMAC128 or KMAC256.

In this section, a KDF specification of KMAC#(K, X, L, S) takes the following parameters.

1. $K - K_I$, the key-derivation key.
2. X − *Context*, a bit string containing the information related to the derived keying material.
3. L − The desired output length of the derived keying material.
4. S − *Label*, an optional customization bit string; for example, *Label* can be an encoding of the characters "KDF" or "KDF4X" in 8-bit ASCII.

Input: K_I, *Context*, L, and *Label*.

Process:

1. If $L > 2^{32} - 1$, output an error indicator and stop (i.e., skip step 2).
2. $K_O = KMAC\#(K_I, Context, L, Label)$.

Output: K_O.

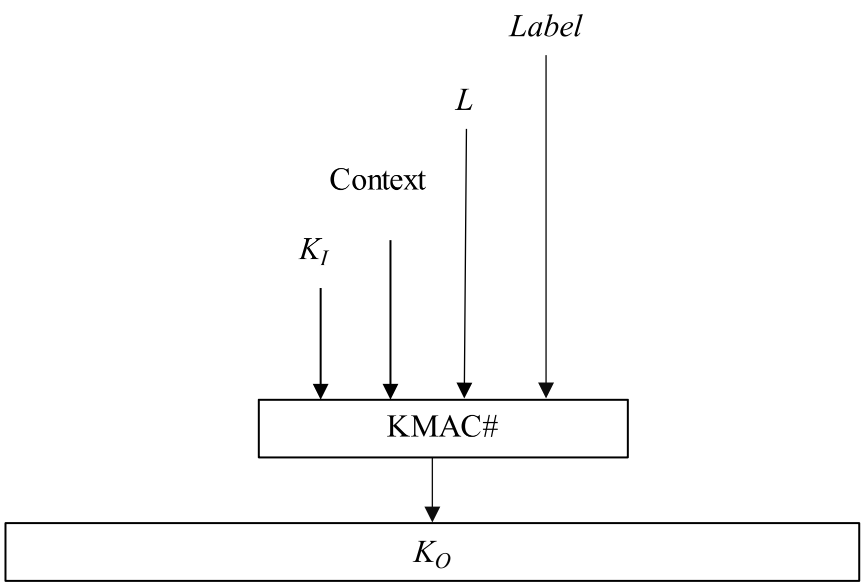

Figure 4. KDF Using KMAC

6 Key Hierarchy

The keying material derived from a given key-derivation key could subsequently be used as one or more key-derivation keys to derive still more key-derivation keys. In this way, a key hierarchy could be established. In a key hierarchy, a KDF is used with a higher-level "parent" key-derivation key (and other appropriate input data) to derive a number of lower-level "child" keys. Figure 5 presents a three-level key hierarchy as an example. In this example, the second level keys $K_I^{(1)}$, $K_I^{(2)}$, and $K_I^{(3)}$ are derived from the top-level key K_I. Assuming that $K_I^{(1)}$, $K_I^{(2)}$, and $K_I^{(3)}$ are used as key-derivation keys, further keys are derived from them as the bottom level keys in the key hierarchy.

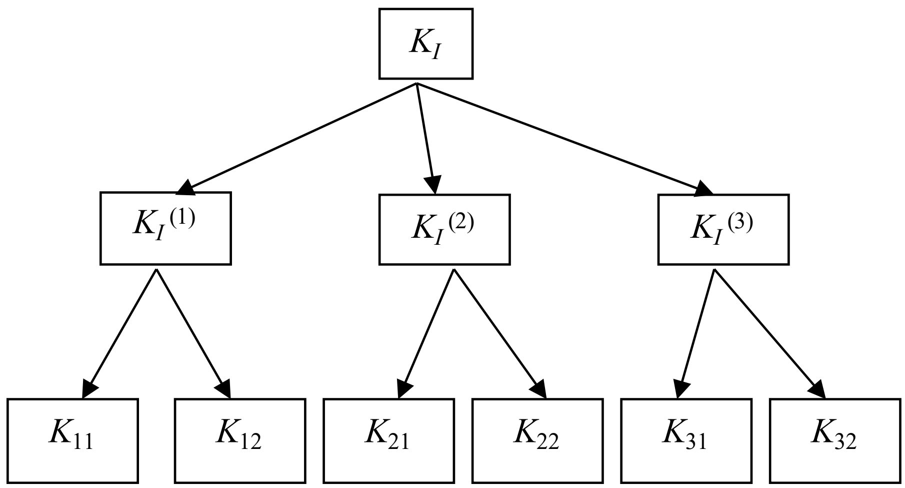

Figure 5. Key Hierarchy

7 Security Considerations

An improperly defined key-derivation function can make the derived keying material vulnerable to attacks. This section will discuss some factors that affect the cryptographic strength of the keying material derived by a KDF. However, some of the required security properties cannot be achieved by the key-derivation function itself. For example, the overall security of the derived keying material depends on the protocols that establish the key-derivation key. These external conditions are out of the scope of the security discussion in this Recommendation.

7.1 Cryptographic Strength

The security strength of a key-derivation function is measured by the amount of work required to distinguish the output of the KDF from a truly uniformly distributed bit string of the same length, under the assumption that the key-derivation key, K_I, is the only unknown input to the KDF. This is certainly no greater than the work required to recover K_I and/or the remaining portions of the derived keying material from a given segment of KDF output. Given a set of input data (other than K_I) and the corresponding output data of sufficient bit length (e.g., no less than w bits), the key K_I can be recovered in (at most) 2^w executions of the KDF, where w is the bit length of K_I through an exhaustive search over all possible K_I values.

7.2 The Length of Key Derivation Key

For some KDFs, the length of the key-derivation key is defined by the PRF used for the key derivation. For example, when using CMAC as a PRF, the key length is uniquely determined by the underlying block cipher. In this case, an implementation **should** check whether the key-derivation key length is consistent with the length required by the PRF.

However, some PRFs can accommodate different key lengths. If HMAC is used as the PRF, then a KDF can use a key-derivation key of essentially any length. It is worth noting, however, that if the chosen key is longer than one input block for the hash function underlying HMAC, that key will be hashed, and the (much shorter) h-bit output will be used as the HMAC key instead. In this case, given a pair consisting of the input data (other than the key) and a sufficient amount of corresponding output of the KDF, the hashed key can likely be recovered in (at most) 2^h computations of the KDF. Therefore, the security strength of an HMAC-based key-derivation function may actually be decreased by increasing the length of the KDK beyond the length of an input block of the underlying hash function.

7.3 Converting Keying Material to Cryptographic Keys

The length of the derived keying material L depends on the requirements of the cryptographic algorithms that rely on the KDF output. The length of a given cryptographic key is determined by the algorithm that will employ it (e.g., a block cipher or a message authentication code) and the desired security strength. In the absence of limitations that may be imposed by relying applications, any segment of the derived keying material that has the required length can be specified for use as a key, subject to the following restriction: when multiple keys (or any other types of secret parameters, such as secret initialization vectors) are obtained from the derived keying material, they **shall** be selected from disjointed (i.e., non-overlapping) segments of the KDF output. Therefore, the value of L **shall** be greater than or equal to the sum of the lengths of the keys and other types of secret parameters that will be obtained from the derived keying material.

Note: To comply with this Recommendation, the derived keying material **shall not** be used as a key stream for a stream cipher[3] (i.e., using the derived keying material to encrypt through an exclusive-or operation with plaintext is not permitted).

7.4 Input Data Encoding

The input data of a key-derivation function consists of different data fields (e.g., a *Label*, the *Context*, and the length of the output keying material). In Section 5, each of the data fields representing certain information is encoded as a bit string. The encoding method **shall** define a one-to-one mapping from the set of all possible input information for that data field to a set of the corresponding bit strings. The different data fields **shall** be assembled in a specific order. The encoding method (including the field order) may be defined in a larger context (e.g., by the protocol that uses a key-derivation function). The encoding method **shall** be designed for unambiguous conversion of the combined input information to a unique bit string.

Unambiguous encoding for input data is required to deter attacks on the KDF that depend on manipulating the input data. For detailed discussions on each attack, see [10].

7.5 Key Separation

In this Recommendation, key separation is a security requirement for the cryptographic keys derived from the same key-derivation key. The keys **shall** be separate in the sense that the compromise of some keys will not degrade the security strength of any of the other keys. In the families of KDFs specified in this Recommendation, key separation can be achieved through different approaches for the following two situations.

1. When keying material for multiple cryptographic keys is obtained from the output of a single execution of a key-derivation function, the segments of the keying material used for the different keys need to be cryptographically separate. The compromise of some keys must not degrade the security of any of the other keys that are obtained from the output of the same execution of a KDF. That is, the compromise of some keys must not make the task of distinguishing any of the other keys from random strings with the same length easier than the task would be if none of the keys were compromised. In order to satisfy this requirement when using the key-derivation functions specified in this Recommendation, different keys **shall** be obtained from disjointed (i.e., non-overlapping) segments of the derived keying material.

2. When keying material for multiple cryptographic keys is obtained from the output of multiple executions of a particular key-derivation function using the same value for K_I, the keying materials output by different calls to the KDF need to be cryptographically

[3] The security strength provided by using the key-derivation functions specified in this Recommendation to generate a key stream for stream ciphers has not been investigated.

separate. The compromise of the keying material output from one of the executions of the KDF must not degrade the security of any of the keying material output from the other executions of the KDF. That is, the compromise must not make the task of distinguishing any of the other keying material from random strings of the same length easier than the task would be if none of the keying material were compromised. In order to satisfy this requirement when using the key-derivation functions specified in this Recommendation, different input data strings (e.g., *Label* || *0x00* || *Context* || $[L]_2$) **shall** be used for different executions. The different data strings can be obtained by including different data related to the derived keying materials. Examples of different information include:

- *Label*, if the keying materials are derived for different purposes.
- Identities included in *Context* if the keying materials are derived for different sets of entities.
- A nonce included in the *Context* if the nonce is communicated by means of the relying protocol and, therefore, shared by each entity who derives the keying material; or
- Session identifiers if the keying materials are derived for different sessions.

7.6 Context Binding

Derived keying material **should** be bound to all relying entities and other information to identify the derived keying material. This is called *context binding*. In particular, the identity (or *identifier*, as the term is defined in [2] and [3]) of each entity that will access (i.e., derive, hold, use, and/or distribute) any segment of the keying material **should** be included in the *Context* string input to the KDF, provided that this information is known by each entity who derives the keying material. In addition to identities, other information related to the derived keying material (e.g., session identifiers, sequence numbers) as well as a nonce may be included in the *Context* string, assuming that the information can be communicated, for instance, by means of the relying protocol.

Context binding may not necessarily increase the security strength of an application making use of a derived key. However, the binding may provide a way to detect protocol errors by providing assurance that all parties who (correctly) derive the keying material are aware of who will access it and in which session it will be used. If those parties have different understandings, then they will derive different keying material. When that keying material is used in a protocol, the protocol will likely fail to complete its execution and, therefore, will indicate errors to the participants.

References

[1] Barker EB, Roginsky AL, Davis R (2020) Recommendation for Cryptographic Key Generation. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-133, Rev. 2. https://doi.org/10.6028/NIST.SP.800-133r2

[2] Barker EB, Chen L, Roginsky AL, Vassilev A, Davis R (2018) Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-56A, Rev. 3. https://doi.org/10.6028/NIST.SP.800-56Ar3

[3] Barker EB, Chen L, Roginsky AL, Vassilev A, Davis R, Simon S (2019) Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-56B, Rev. 2. https://doi.org/10.6028/NIST.SP.800-56Br2

[4] Barker EB, Kelsey JM (2015) Recommendation for Random Number Generation Using Deterministic Random Bit Generators. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-90A, Rev. 1. https://doi.org/10.6028/NIST.SP.800-90Ar1

[5] Barker EB, Chen L, Davis R (2020) Recommendation for Key-Derivation Methods in Key-Establishment Schemes. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-56C, Rev. 2. https://doi.org/10.6028/NIST.SP.800-56Cr2

[6] Dworkin MJ (2005) Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-38B, Includes updates as of October 6, 2016. https://doi.org/10.6028/NIST.SP.800-38B

[7] National Institute of Standards and Technology (2008) The Keyed-Hash Message Authentication Code (HMAC). (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) 198-1. https://doi.org/10.6028/NIST.FIPS.198-1

[8] Kelsey JM, Chang S-jH, Perlner RA (2016) SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-185. https://doi.org/10.6028/NIST.SP.800-185

[9] Goldreich O, Goldwasser S, Micali S (1986) How to construct pseudorandom functions, *Journal of the ACM*. Vol. 33, No. 4, pp. 210-217. https://doi.org/10.1145/6490.6503

[10] Adams C, Kramer G, Mister S, Zuccherato R (2004) On the Security of Key Derivation Functions. *Information Security* (Springer Verlag), LNCS 3225, pp. 134-145. http://doi.org/10.1007/978-3-540-30144-8_12

Appendix A—Revisions

The original version of this document was published in November 2008. In October 2009, the publication was updated with the following change:

In Section 5, page 12, at the end of the paragraph, “For each of the iterations of the PRF, the key derivation key K_I is used as the key, and the input data consists of an iteration variable and a string of fixed input data. Depending on the mode of iteration, the iteration variable could be a counter, the output of the PRF from the previous iteration, a combination of both, or an output from the first pipeline iteration in the case of double-pipeline iteration mode. In the following key derivation functions, the fixed input data is a concatenation of a Label, a separation indicator 0x00, the Context, and $[L]_2$…”, the sentence “One or more of these fixed input data fields may be omitted unless required for certain purposes, as discussed in Section 7.5 and Section 7.6.” was added.