NIST Special Publication 800-108

Recommendation for Key Derivation Using Pseudorandom Functions

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 2008

U.S. Department of Commerce
Carlos M. Gutierrez, Secretary

National Institute of Standards and Technology
Patrick Gallagher, Deputy Director

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.

KEY WORDS: key derivation, pseudorandom function

Acknowledgements

The author, Lily Chen of the National Institute of Standards and Technology (NIST), would like to thank her colleagues, Elaine Barker, William Burr, Quynh Dang, Donna Dodson, Morris Dworkin, Katrin Hoeper, Jim Nechvatal, Tim Polk, Allen Roginsky of NIST, and Rich Davis of National Security Agency, for helpful discussions and valuable comments.

The author also gratefully appreciates the thoughtful and instructive comments received during the public comment period, 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 key derivation schemes, as specified in this Recommendation, will be conducted within the framework of the Cryptographic Module Validation Program (CMVP), a joint effort of NIST and the Communications Security Establishment Canada. An implementation of a key derivation function must adhere to the requirements in this Recommendation in order to be validated under the CMVP. The requirements of this Recommendation are indicated by the word “shall.”

Table of Contents

Figures

1. Introduction

When parties share a secret symmetric key (e.g., upon a successful execution of a key-establishment scheme as specified in [1] and [2]), it is often the case that additional keys will be needed (e.g. as described in [3]). Separate keys may be needed for different cryptographic purposes – for example, one key may be required for an encryption algorithm, while another key is intended for use by an integrity protection algorithm, such as a message authentication code. At other times, the distinct keys required by multiple entities may be generated by a trusted party from a single master key. Key derivation functions are used to derive such keys.

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 a key that has been established through an automated key-establishment scheme (e.g. as defined in [1] and [2]), or from a pre-shared key (e.g., a manually distributed key).

Effectively, the key derivation functions specified in this Recommendation provide the key expansion functionality described in [4], 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).

Note that the key-agreement schemes specified in [1] and [2] already incorporate the use of a (hash-based) key derivation function. If the key used as an input to one of the key derivation functions specified in this Recommendation has been established by using one of those key-agreement schemes, then, for all intents and purposes, that key has been obtained by employing one of the key derivation functions defined in [1] and [2] as a randomness extractor.

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.
Cryptographic key	A binary string used as a secret parameter by a cryptographic algorithm. In this Recommendation, a cryptographic key **shall** be either a truly random binary string of a length specified by the cryptographic algorithm or a pseudorandom binary string of the specified length that is computationally indistinguishable from one selected uniformly at random from the set of all binary strings of that length.
Entity	An individual (person), organization, device or a combination thereof. "Party" is a synonym. 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 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 [6].
Key derivation	The process that derives keying material from a key.
Key derivation function	A function that, with the input of a cryptographic key and other data, generates a binary string, called keying material.
Key derivation key	A key used as an input to a key derivation function to derive other keys.
Key-establishment	A procedure, conducted by two or more participants, after which the resultant keying material is shared by all participants.
Key hierarchy	A key hierarchy is 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 binary string, such that any non-overlapping segments of the string with the required lengths can be used as symmetric cryptographic keys.
Message authentication code	A family of cryptographic algorithms that is parameterized by a symmetric key. Each of the algorithms can act on input data of 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.
Pipeline	A term used to describe a series of sequential PRF executions.
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.

Security strength	A measure of the computational complexity associated with recovering certain secret and/or security-critical information concerning a given cryptographic algorithm from known data (e.g. plaintext/ciphertext pairs for a given encryption algorithm). In this Recommendation, the security strength of a key derivation function is measured by the work required to distinguish the output of the KDF from a bit string selected uniformly at random from the set of all bit strings with the same length as the output of the KDF, under the assumption that the key derivation key is the only unknown input to the KDF.
Shall	This term is used to indicate a requirement of a Federal Information Processing Standard (FIPS) or 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

$A(i)$	The output of the i^{th} iteration in the first pipeline in a double pipeline iteration mode.
$A \| B$	The concatenation of binary strings A and B.
CMAC	Cipher-based Message Authentication Code (as specified in NIST SP 800-38B [7]).
h	An integer whose value is the length of the output of the PRF in bits.
H()	A hash function.
HMAC	Keyed-hash Message Authentication Code (as specified in FIPS 198-1 [8]).
i	The counter for each iteration, which is represented as a binary string of length r when it is an input to each iteration of the PRF.
IV	A binary string that is used as an initial value in computing the first iteration 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. For a key derivation, K_I is used (along with other data) to derive keying material K_O.
K_O	Keying material that is derived from a key derivation key K_I and other data.

L	An integer specifying the length of the derived keying material K_O in bits, which is represented as a binary 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, smaller or equal to 32, 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.
$\|S\|$	The length (in bits) of binary string S.
$[T]_2$	An integer T represented as a binary string (denoted by the "2") with a length specified by the function, an algorithm, or a protocol which uses T as an input.
w	An integer that denotes 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.
$\varnothing$	The empty binary string. That is, for any binary string A, $\varnothing \| A = A \| \varnothing = A$.
0x00	An all zero octet.

4. Pseudorandom Function (PRF)

A pseudorandom function is the basic building block in constructing a key derivation function in this Recommendation. Generally, a pseudorandom function family $\{PRF(s, x) \mid s \in S\}$ consists of polynomial time computable functions with an index (also called a seed) s and input variable 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, please 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 either the keyed-

hash Message Authentication Code (HMAC) specified in [8] or the cipher-based Message Authentication Code (CMAC) specified in [7] as the pseudorandom function. For a given KDF using HMAC or CMAC, the key K_I is assumed to be computationally indistinguishable from one that has been selected uniformly at random from the set of all the binary strings with length of $|K_I|$.

Note that [1] and [2] specify additional (hash-based) key derivation functions that are tailored to the needs of the key establishment schemes described in those documents.

5. Key Derivation Functions (KDF)

This Section defines several families of key derivation functions that use PRFs. For the purposes of this Recommendation, a key derivation function (KDF) is a function with which an input key and other input data are 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. To comply with this Recommendation, a key derivation key **shall** be a cryptographic key (see Section 3.1). The key derivation key used as an input to one of the key derivation functions specified in this Recommendation can be generated by an approved cryptographic random bit generator (e.g., by a deterministic random bit generator of the type specified in [5]), or by an approved automated key-establishment process (e.g., as defined in [1] and [2]).

When the key derivation key is established through an automated key-establishment process, the key derivation key **shall** be a segment of the secret keying material that results from that process. The nomenclature secret keying material is intended to distinguish the (pseudo)random binary strings output by such a process from various precursors, such as shared secret values computed through algebraic operations on the public and private values in a Diffie-Hellman key-agreement scheme (e.g., $g^{xy} \bmod p$).

Any disjoint segments of the derived keying material (with the required lengths) can be used as cryptographic keys for the corresponding algorithms. However, in order to make sure that different parties will obtain the same keys from the derived keying material, the cryptographic scheme 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 scheme 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.

The KDFs specified in this Recommendation are constructed using PRFs (see Section 4). Depending on the intended length of the keying material to be derived, the KDF may require multiple invocations of the PRF. A way to iterate the multiple invocations is called a mode of iteration. In this Recommendation, a counter mode is specified in Section 5.1, a feedback mode in Section 5.2, and a double-pipeline iteration mode in Section 5.3. In keeping with the terminology above, the output of a key derivation function is called the derived keying material and may subsequently be segmented into multiple keys.

To define key derivation functions, the following notations are used. Some of the notations have been defined in Section 3.2. They 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 key, and the other input data is used as the text as defined in [8]. 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 as defined in [7].

2) K_O – Keying material output from a key derivation function specified in this Recommendation, a binary string of the required length, which is derived using a key derivation key.

3) *Label* – A string that identifies the purpose for the derived keying material, which is encoded as a binary 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 binary string containing the information related to the derived keying material. It may include identities of 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 binary 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.

6) L – An integer specifying the length (in bits) of the derived keying material K_O. L is represented as a binary string when it is an input to a key derivation function. The length of the binary 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 binary string of length r that is an input to each iteration of a PRF in counter mode and (optionally) in feedback mode and double-pipeline iteration mode.

10) r – An integer, smaller or equal to 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 used to indicate a separation of different variable length data fields[1].

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

A key derivation function iterates a pseudorandom function n times and concatenates the outputs until L bits of keying material are generated, where $n = \lceil L/h \rceil$. For counter mode, n **shall not** be larger than 2^r-1, where $r \leq 32$ is the binary length of the counter. For feedback mode and double-pipeline iteration mode, n is limited to 2^{32}-1 in this section based on the fact that $L = (2^{32}\text{-}1)\ h$ bits keying material is far more than enough for most applications. Notice that in these two modes, even though the counter is used as an input, it does not affect this limit. However, for each KDF or an application which uses a KDF, a smaller limit can be applied to the number of iterations.

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 length for each data field and an order **shall** be defined unambiguously. For example, the length and the order may be defined as a 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 variable. The mode is defined as follows.

Fixed values:

1. h - The length of the output of the PRF in bits, and
2. 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 indicate an error and stop.
3. `result(0):=` $\varnothing$.
4. For i = 1 to n, do

   a. `K(i)` := PRF (K_I, $[i]_2$ || *Label* || *0x00* || *Context* || $[L]_2$)

   b. `result(i) := result(i-1) || K(i).`

5. Return: K_O := the leftmost L bits of `result(n)`.

Output: K_O.

In each iteration, the fixed input data is the string *Label* || *0x00* || *Context* || $[L]_2$. The counter $[i]_2$ is the iteration variable and is represented as a binary string of r bits. The KDF in counter mode is illustrated in Figure 1.

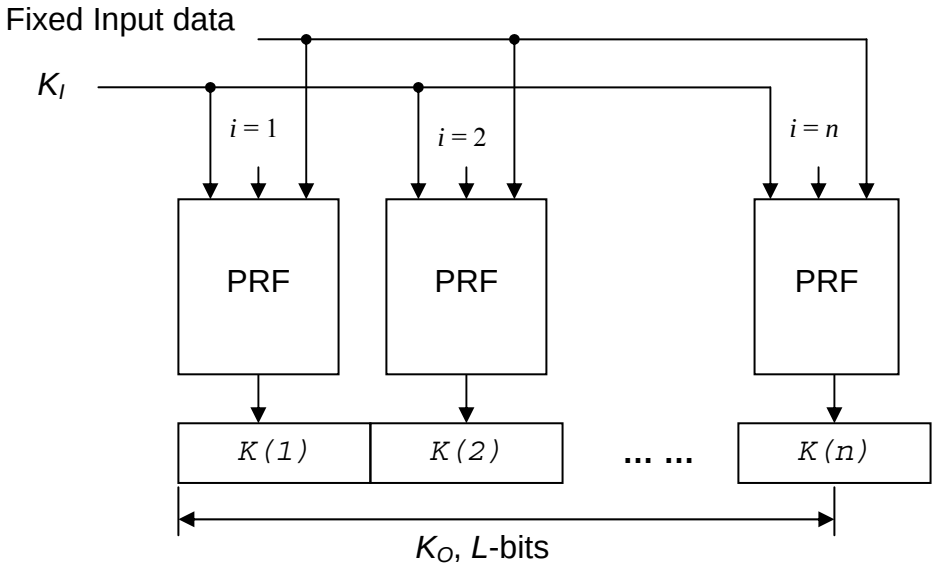

Figure 1: KDF in Counter Mode

5.2 KDF in Feedback Mode

This section specifies a family of KDFs that uses the feedback mode. In feedback mode, the output of the PRF is computed using the result of the previous iteration and, optionally, using a counter as the iteration variable(s). The mode is defined as follows. (Note that when $L \leq h$, `IV` = ∅, 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.)

Fixed values:

1. h - The length of the output of the PRF in bits, and
2. 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, then indicate an error and stop.
3. `result(0):=` ∅ and `K(0):= IV`.

4. For $i = 1$ to n, do

   a. $K(i) := \text{PRF}(K_I, K(i-1) \{|| [i]_2 \}|| Label || 0x00 || Context || [L]_2)$

   b. $result(i) := result(i-1) || K(i)$

5. **Return**: $K_O :=$ the leftmost L bits of $result(n)$.

Output: K_O.

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

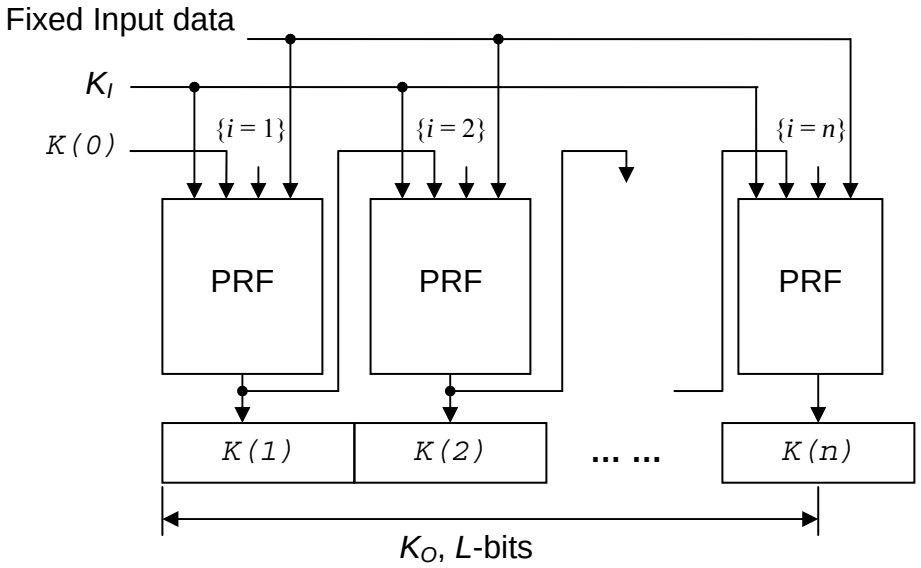

Figure 2: KDF in Feedback Mode

5.3 KDF in Double-Pipeline Iteration 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.

Fixed values:

1. h - The length of the output of the PRF in bits, and
2. 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, then indicate an error and stop.
3. `result(0)` := ∅
4. `A(0)` := **`IV`** = *Label* || *0x00* || *Context* || $[L]_2$.
5. For i = 1 to n, do
   a. `A(i)` := PRF (K_I, `A(i-1)`)
   b. `K(i)` := PRF (K_I, `A(i)` {|| $[i]_2$}|| *Label* || *0x00* || *Context* || $[L]_2$)
   c. `result (i)` := `result(i-1) || K(i)`
7. Return: K_O, i.e., the leftmost L bits of `result(n)`.

Output: K_O.

The first iteration pipeline uses a feedback mode with an initial value of `A(0)= IV =` *Label* || *0x00* || *Context* || $[L]_2$. Each second pipeline iteration generates `K(i)` using `A(i)` and, optionally, a counter $[i]_2$ as the iteration variable. The KDF in the double-pipeline iteration mode is illustrated in Figure 3.

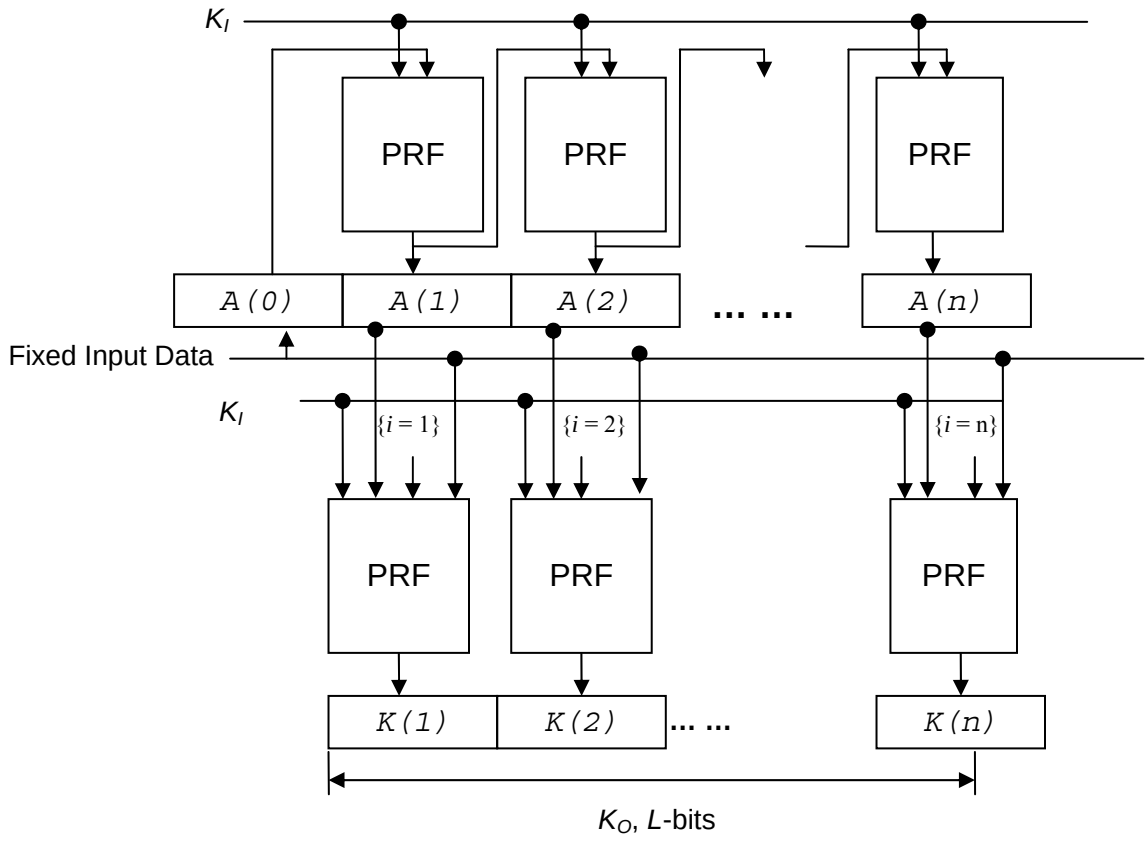

Figure 3: KDF in Double-Pipeline Iteration Mode

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 4 presents a three-level key hierarchy as an example. In the hierarchy described by Figure 4, the second level keys $K_I^{(1)}$, $K_I^{(2)}$, and $K_I^{(3)}$ are derived from the top level key K_I. Assuming $K_I^{(1)}$, $K_I^{(2)}$, and $K_I^{(3)}$ are used as key derivation keys, from each of them, further keys are derived as the bottom level keys in the key hierarchy.

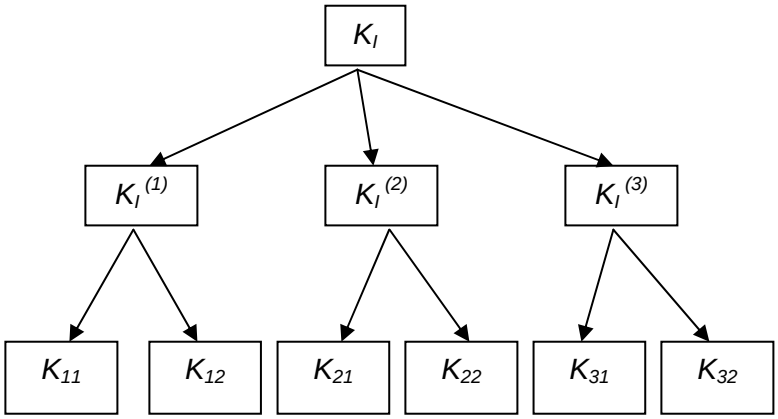

Figure 4: Key Hierarchy

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 amount 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), 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 the 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 the HMAC is used as the PRF, then a KDF can use a key derivation key of essentially any length. It is worth noting that when the key length is longer than the block length of the underlying hash function for HMAC, the key will be hashed to h bits first, where h is the length of the hash function output. In this case, given a pair of the input data and the corresponding output value of the PRF, the hashed key can be recovered in (at most) 2^h computations of the PRF. Therefore, the security strength may not be increased with a longer key length.

7.3 Converting Keying Material to Cryptographic Keys

The length, L, of the derived keying material is dependent upon 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 – for example, 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 having 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, e.g. secret initialization vectors) are obtained from the derived keying material, they **shall** be selected from disjoint (i.e., non-overlapping) segments of the KDF output. Therefore, the value of L **shall** be equal to or greater than the sum of the lengths of the keys (etc.) that will be obtained from the derived keying material.

In Section 5.1, n, the number of iterations of the PRF computations, is limited by 2^r-1, where $r \leq 32$ is the binary length of the counter that is used as input during the iterations. This ensures that the counter values $[i]_2$ used as an input to the PRF will not repeat during a particular call to the KDF function. This also limits the length of the derived keying material to $L \leq (2^r-1)h$, where h is the bit length of the PRF output. For feedback mode and double-pipeline iteration mode, the number of iterations is limited to $2^{32} - 1$.

To comply with this Recommendation, the derived keying material **shall not** be used as a key stream for a stream cipher[2].

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 binary string. The encoding method **shall** define a one-to-one mapping from the set of all possible input information

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

for that data field to a set of the corresponding binary 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, for example, 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 binary 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, please 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 by different keys need to be cryptographically separate in the following sense: 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 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** use disjoint (i.e. non-overlapping) segments of the derived keying material.
2. When keying materials 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 separate in the following sense: 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 through including different information related to the derived keying materials. Examples of different information include
    - *Label*, if the keying materials are derived for different purposes;
    - identities in *Context*, if the keying materials are derived for different sets of entities;

- a nonce in *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 [1] and [2]) of each entity that will access (meaning 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. Besides identities, other information related to the derived keying material such as session identifiers, sequence numbers, as well as nonce may be included in the *Context* string, assuming 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.

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] IETF RFC 5216, The EAP-TLS Authentication Protocol, March 2008.

[4] Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk, and T. Rabin, Randomness Extraction and Key derivation Using the CBC, Cascade, and HMAC Modes, Crypto'04, LNCS 3152, pp. 494-510. Springer Verlag, 2004.

[5] NIST SP 800-90, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, March 2007.

[6] FIPS 180-3, Secure Hash Standard, Revision expected to be published in 2008.

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

[8] FIPS 198-1, The Keyed-Hash Message Authentication Code (HMAC), Revision expected to be published in 2008.

[9] O.Goldreich, S. Goldwasser and S. Micali, "How to construct pseudorandom functions", Journal of the ACM, Vol. 33, No. 4, pp. 210-217, (1986).

[10] C. Adams, G. Kramer, S. Mister, and R. Zuccherato "On the Security of Key Derivation Functions", Information Security, LNCS 3225, pp. 134-145, Springer Verlag, 2004.