INTERNATIONAL STANDARD

ISO/IEC 11770-6

First edition
2016-10-01

Information technology — Security techniques — Key management —

Part 6: Key derivation

Technologies de l'information — Techniques de sécurité — Gestion de clés —

Partie 6: Dérivation de clés

Reference number
ISO/IEC 11770-6:2016(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

A list of all parts in the ISO/IEC 11770 series can be found on the ISO website.

Introduction

The establishment of shared secret cryptographic keys is a fundamental key management service. It is a prerequisite for the use of a range of symmetric cryptographic techniques, including symmetric encryption for confidentiality protection, and message authentication codes (MACs) for integrity protection and data origin authentication. Key derivation techniques enable such keys to be generated from pre-existing secrets and have a range of possible applications. Two particularly important applications are as follows.

First, while two (or more) parties might share secret information, this secret information might not be suitable for immediate use as input to an encryption algorithm or a message authentication code scheme. For example, the initial secret information might not be distributed randomly across the entire space of possible values, or an unauthorized third party might have partial information about it. A key derivation function (or a key extraction function) can be used to resolve this issue by taking the secret information as input, perhaps together with other non-secret material, and giving a suitable secret key as output.

Second, a number of secret keys might be required for different purposes, e.g. for different applications or for input to different cryptographic functions. Again, a key derivation function (or a key expansion function) can be used to meet this requirement by taking secret information, perhaps together with other non-secret material, as input, and giving a secret key, or keys, as output. The secret information might, for example, be shared by two or more parties, and the generated secret keys could then be used to protect data exchanged between these parties via untrusted channels; alternatively, the secret information might only be known by a single party, and the generated keys could then be used to protect data stored by that party in untrusted locations.

This document is concerned with such key derivation techniques. Two general classes of key derivation techniques are specified, namely one-step and two-step functions, both of which can be used to generate either a single key or multiple keys. One-step functions transform the input information into one or more keys in a single operation. Two-step functions first transform the input information into a secret MAC key, which is then used in the second step (which can be executed multiple times) to generate one or more secret keys for use in applications.

The choice between one-step and two-step functions depends on two main things: the nature of the available secret input to the key derivation function, and the way in which the secret input is to be used. For example, if the available secret input is already in the form of a secret key, then a one-step function will normally be appropriate. Also, regardless of the nature of the secret input, if the function is to be used only once with a particular set of secret inputs, then again a one-step function will typically be appropriate. However, if the secret input is not in the form of a secret key, and the same secret input is to be used multiple times to generate one or more keys, then a two-step function is likely to be appropriate, where the first step is performed once to generate a MAC key and the second step is performed whenever a new key is, or keys are, to be generated from the MAC key.

This document defines a range of one-step key derivation functions. It also defines examples of both key extraction functions and key expansion functions, where a key extraction function can be combined with a key expansion function to define a two-step key derivation function.

Information technology — Security techniques — Key management —

Part 6: Key derivation

1 Scope

This document specifies key derivation functions, i.e. functions which take secret information and other (public) parameters as input and output one or more "derived" secret keys. Key derivation functions based on MAC algorithms and on hash-functions are specified.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 9797 (all parts), *Information technology — Security techniques —Message Authentication Codes (MACs)*

ISO/IEC 10118 (all parts), *Information technology — Security techniques — Hash-functions*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 11770-1 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

3.1
entropy
measure of the disorder, randomness or variability in a closed system

Note 1 to entry: The particular measure of entropy that is used in the document is discussed in 5.1.

3.2
hash-function
function which maps strings of bits of variable (but usually upper bounded) length to fixed-length strings of bits

Note 1 to entry: Cryptographic requirements on hash-functions employed for the purposes of this document are considered in 6.1.

3.3
key derivation function
KDF
function which takes as input a number of parameters, at least one of which shall be secret, and which gives as output keys appropriate for the intended algorithm(s) and applications

Note 1 to entry: Cryptographic requirements on key derivation functions are specified in 5.1.

Note 2 to entry: Key derivation functions are also sometimes known as "key generating functions."

3.4
key expansion function
KPF
function which takes as input a number of parameters, at least one of which shall be a secret key, and which gives as output keys appropriate for the intended algorithm(s) and applications

Note 1 to entry: Cryptographic requirements on key expansion functions are specified in 7.1.

Note 2 to entry: All the KPFs specified in this document are based on a MAC algorithm.

3.5
key extraction function
KTF
function which takes as input a number of parameters, at least one of which shall be secret, and which gives as output a key suitable for use as input to a key expansion function

Note 1 to entry: Cryptographic requirements on key extraction functions are specified in 7.1.

3.6
message authentication code
MAC
string of bits which is the output of a MAC algorithm

[SOURCE: ISO/IEC 9797-1:2011, 3.9 – modified, Note 1 to entry removed]

3.7
message authentication code algorithm
MAC algorithm
algorithm for computing a function which maps strings of bits and a secret key to fixed-length strings of bits, satisfying the following two properties:

— for any key and any input string, the function can be computed efficiently;

— for any fixed key, and given no prior knowledge of the key, it is computationally infeasible to compute the function value on any new input string, even given knowledge of a set of input strings and corresponding function values, where the value of the ith input string might have been chosen after observing the value of the first $i - 1$ function values (for integers $i > 1$)

Note 1 to entry: A MAC algorithm is sometimes called a "cryptographic check function."

Note 2 to entry: Computational feasibility depends on the user's specific security requirements and environment.

Note 3 to entry: Additional cryptographic requirements for MAC algorithms employed for the purposes of this document are specified in 6.1 and 7.1.

3.8
message authentication code key
MAC key
bit string suitable for use as a key input to a MAC algorithm

3.9
one-step key derivation function
OKDF
key derivation function which operates in a single stage, in contrast to key derivation functions involving separate key-extraction and key-expansion stages (cf. 3.12)

3.10
salt
value used as input to a key derivation function, a key expansion function or a key extraction function, which might not be a secret

3.11
secret information
bit string used as input to a KDF or a KTF, which shall be known only to entities which are authorized to agree upon the key or keys, and possibly to one or more other entities trusted for the purposes of key establishment

3.12
two-step key derivation function
TKDF
key derivation function which involves two stages, namely the use of a key extraction function followed by a key expansion function

4 Symbols and abbreviations

4.1 Symbols

a	algorithm identifier
b	bit string output by a KDF
c	counter
f	MAC algorithm, where $f_k(d)$ denotes the MAC obtained when f is given as input the key k and the data d (a bit string)
h	hash-function
k	secret key
k_m	secret MAC key
L_b	bit-length of the output b of a KDF
L_c	bit-length of the binary encoding of the counter c
L_f	bit-length of the output of the MAC algorithm f used as part of a KDF
L_h	bit-length of the output of a hash-function h used as part of a KDF
L_k	bit-length of a secret key k
p	label employed to 1) indicate how a bit string is to be partitioned into one or more keys, and 2) identify the algorithm(s) with which the resulting key(s) should be used
s	secret information
t	salt value

u auxiliary secret information

y bit string

z bit string

4.2 Abbreviations

KDF key derivation function

KPF key exPansion function

KTF key exTraction function

MAC message authentication code

OKDF one-step KDF

TKDF two-step KDF

4.3 Notation

|| concatenation, where $x||y$ denotes the bit string obtained by concatenating the bit strings x and y in the order specified

$\lceil \ldots \rceil$ if x is a real number, $\lceil x \rceil$ denotes the smallest integer greater than or equal to x

[...] if x is a bit string, $[x]$ indicates that the inclusion of x is optional

5 Key derivation techniques

5.1 Model

A KDF is a function which can be used to generate one or more secret keys, given a range of possible inputs. In general, a KDF has the following inputs and outputs.

Inputs:

— Mandatory inputs:

  — s secret information possessed by the party or parties using the KDF. It shall be computationally infeasible for parties not authorized to obtain the keys output from the KDF to differentiate between values produced using the mechanism employed to generate s and values selected with sufficient entropy to support the security requirements of the user(s) of the intended application(s) of the generated key(s) (see B.1.2 for guidance).

NOTE 1 The entropy of a (discrete) random variable X is a mathematical measure of the uncertainty associated with the possible outcomes of an observation, which serves as an estimate of the information that may be gained from an observation of X. Entropy is at its highest if all possible outcomes are equally likely. Entropy is lessened if the distribution on X is such that some outcomes are more likely than others; entropy is at its lowest (zero) if each observation is certain to be a particular (predetermined) value.

NOTE 2 There are various notions of entropy that are relevant to cryptographic applications, but in this document entropy is measured using min-entropy, as defined in ISO/IEC 18031:2011, 3.22[5]. When the unit of measurement is bits, the min-entropy of a (discrete) random variable X is the largest value m for which the probability of observing any particular value of X is at most $(1/2)^m$. The min-entropy of X is a lower bound for other commonly used measures of entropy.

NOTE 3 Computational feasibility depends on the user's specific security requirements and environment.

— Optional/mechanism-dependent inputs:

  — *a* algorithm identifier employed by the OKDF2 mechanism, which specifies the algorithm with which the single key it gives as output should be used;

  — *p* label, used with certain KDFs to indicate how the output bit string should be partitioned into secret keys and the intended use for each such secret key;

  — *t* salt value, i.e. a (not necessarily secret) value that can be used, for example, to make the generated key(s) specific to a particular application and/or to ensure that the output key(s) will vary with each execution of a key derivation function; used as a MAC key by the KTF1 mechanism;

  — *t′* salt value, i.e. a (not necessarily secret) value that is used as a MAC key by the OKDF6 mechanism and used as an initialization value by the KPF3 mechanism;

  — *u* auxiliary secret information possessed by the party or parties using the KDF; the entropy provided by the choice of *u* may be used to supplement that provided by the choice of *s*, in support of the security requirements of the user(s) of the intended application(s) of the generated key(s).

Outputs:

— *b* bit string that is either equal to a single secret key or can be partitioned to obtain a number of secret keys.

The instantiation of a key derivation function specified in this document requires a number of critical choices, including the selection of a hash-function or MAC algorithm(s), the type and source of various input parameters, etc. Appropriate choices will provide assurance that the resulting KDF satisfies the following requirement.

Given no prior knowledge of *s*, it shall be computationally infeasible to predict the value of an (as-yet-unseen) output *b* with a probability of success that is a significant improvement over simply guessing either the value of *b* or the value of *s*. This shall be the case even when given knowledge of a (bounded) number of KDF outputs which were computed using the same (unknown) *s*, but with other (possibly known and adaptively chosen) input parameters that are not identical to those used to generate *b*.

NOTE 4 Computational feasibility and the significance of the probability of successfully predicting *b* depend on the KDF user's specific security requirements and environment.

5.2 Types of key derivation function

In this document, two types of KDF are specified.

— **One-step KDFs (OKDFs)** are KDFs which transform input information into one or more keys in a single operation.

— **Two-step KDFs (TKDFs)** are KDFs which transform input information into one or more keys in two operations. A TKDF consists of a combination of two functions: a key extraction function (KTF), and a key expansion function (KPF). In some cases, the KTF can be the identity function (i.e. a function whose output is always the same as the input), i.e. the TKDF is simply a KPF, in which case the secret information input to the TKDF shall take the form of a MAC key suitable for use with the MAC algorithm on which the KPF is based.

Six OKDFs (OKDF1 to OKDF6) are specified in Clause 6. In Clause 7, one key extraction function (KTF1), four key expansion functions (KPF1 to KPF4), and four TKDFs (TKDF1 to TKDF4) are specified, where the TKDFs are defined as specific combinations of a KTF and a KPF.

The KDFs specified in this document are built upon existing cryptographic functions, namely either hash-functions, as specified in ISO/IEC 10118 (all parts), or message authentication code algorithms, as specified in ISO/IEC 9797 (all parts).

5.3 Relationship to key management life cycle

ISO/IEC 11770-1:2010, Clause 4[1] specifies a model for key management, including, in 4.3, a model for the life cycle of a key. Key derivation forms part of the first step in this life cycle, namely the "Generation" step. Key derivation could also be used in the "Active" state for a key, when an existing active key is used to generate additional keys.

5.4 Use of a key derivation function

The inputs to and outputs from a KDF are permitted to take a variety of forms.

The secret information *s* input to a KDF could be an existing secret key, secret information established between two parties, e.g. using one of the key establishment mechanisms specified in ISO/IEC 11770-2[2] or ISO/IEC 11770-3[3], or any other secret data known by the party or parties using the KDF.

A KDF could be invoked once with particular secret information *s* to generate either a single secret key or to generate a number of secret keys. Alternatively, it could be invoked multiple times with the same *s* and different salt values *t* to generate a series of secret keys for specific applications. These salt values could take a range of forms and have varying lengths, even when used with the same secret value *s*.

Guidance on use is provided in Annex B.

6 One-step key derivation functions

6.1 General

A one-step KDF (an OKDF) is a KDF which is capable of transforming input information into one or more secret keys in a single phase of operations (in contrast to the separate extraction and expansion phases that are employed by two-step KDFs). Like all KDFs, an OKDF takes as input secret information together with other parameters, some or all of which might be public, and gives as output one or more secret keys.

In Clause 6, six OKDFs (OKDF1 to OKDF6) are specified. Five of them (OKDF1 to OKDF5) are based on the use of a cryptographic hash-function chosen from ISO/IEC 10118 (all parts). The other scheme (OKDF6) is based on the use of a MAC algorithm selected from ISO/IEC 9797 (all parts). OKDF2 outputs a single secret key, whereas the other five functions output a bit string which can be partitioned into multiple secret keys, as required.

A hash-function used by OKDF1 to OKDF5 shall satisfy the following three properties:

— it is computationally infeasible to construct two distinct inputs that map to the same output (collision resistance);

— for a given output, it is computationally infeasible to find an input that maps to this output (preimage resistance);

— for a given input, it is computationally infeasible to find a second (distinct) input that maps to the same output (second preimage resistance).

The hash-function employed in an implementation of OKDF1 to OKDF5 and/or the method(s) used to encode the data input to that hash-function shall be chosen in a manner that guards against length-extension attacks. A hash-function *h* is called "length-extension resistant" if, given the hash value, $h(x)$, of some unknown input *x* (and possibly given the length of *x* as well), it is computationally infeasible to deduce (with non-negligible probability of success) the value $h(x||y)$ for any non-empty bit string *y*, unless *x* itself can be guessed. Examples of hash-functions that are length-extension resistant include SHA-384 and SHA-512/256, which output truncated versions of a 512-bit hash value. Length-extension resistance is not required of *h* in applications where each bit string used as input to the hash-function has the same fixed length, or in situations where the input strings correspond to information that has been encoded in a prefix-free fashion, ensuring that no (non-empty) input string will appear as the initial segment of another (longer) input string.

In OKDF6, a MAC function f is used with a MAC key that may be publicly known, but with secret information included as part of the input message data. The MAC algorithm employed in an implementation of OKDF6 shall satisfy the following property, in addition to those that are ordinarily required (cf. 3.7).

Given that the key k input to the MAC function f is a publicly known value, and the input data d includes secret information s, then, without prior knowledge of s, it shall be computationally infeasible to predict the (as-yet-unseen) MAC value $f_k(d)$ with a probability of success that is a significant improvement over simply guessing either $f_k(d)$ or s. This shall be the case even when given knowledge of a (bounded) number of MAC values $f_{kj}(d_j)$, where each d_j includes the same (unknown) s and where both k_j and the non-secret portions of d_j are known and (adaptively) chosen, subject to the restriction that (k_j, d_j) is not equal to (k, d).

6.2 One-step key derivation function 1 (OKDF1)

6.2.1 General

This key derivation function takes as input secret information s and (optionally) a salt t, which does not need to be a secret. It gives as output a bit string b which can be partitioned into one or more secret keys. If the key derivation function is invoked multiple times with the same s to generate a series of distinct secret keys, then a different value of t shall be used with each invocation.

NOTE The "IEEE P1363 key derivation function" mechanism described in ISO/IEC 11770-3[3] is a special case of the OKDF1 mechanism, where s and t are encoded as octet strings (with t representing the key derivation parameters) and the output bit string is used as a single secret key.

6.2.2 Requirements for use

Prior to use, the one or more entities which wish to use the function shall meet the following requirements.

— They shall agree on the use of OKDF1.

— They shall agree on a specific hash-function h, which shall be one of the algorithms specified in ISO/IEC 10118 (all parts). This hash-function shall give as output a string of bit-length greater than or equal to the sum of the bit-lengths of the keys to be generated.

— They shall agree on the encoding methods for the strings s and (if used) t prior to concatenation in step a) of 6.2.3.

— They shall agree on precisely how the output of the KDF shall be partitioned into secret keys.

— If the option to employ a salt is exercised, they shall agree on a specific salt value t, which does not need to be a secret.

— They shall all possess a bit string s, constituting the secret information.

6.2.3 Operation of function

The function involves the following two steps.

a) Compute $b = h(s \,||\, [t])$.

b) Partition b into one or more secret keys.

6.3 One-step key derivation function 2 (OKDF2)

6.3.1 General

This key derivation function takes as input secret information s, an algorithm identifier a, (optionally) a salt t, which does not need to be a secret, and (optionally) auxiliary secret information u. It gives as output a single secret key k for use with a specific algorithm. If the key derivation function is invoked multiple times with the same s to generate a series of distinct secret keys, then a different value of a, t, or u shall be used with each invocation.

NOTE Provided that its output is used as a single secret key, an instance of the "ANSI X9.42 key derivation function" mechanism described in ISO/IEC 11770-3[3] could be viewed as an example of the OKDF2 mechanism, where the counter c is encoded as a 32-bit octet string, t and u (if used) comprise an appropriate subset of the (optional) key-specification information denoted by EntityAInfo, EntityBInfo, SuppPrivInfo and SuppPubInfo in ISO/IEC 11770-3[3], and the quantity denoted $a \,||\, c \,||\, [t] \,||\, [u]$ in the specification of OKDF2 is encoded using an ASN.1 DER encoding.

6.3.2 Requirements for use

Prior to use, the one or more entities which wish to use the function shall meet the following requirements.

— They shall agree on the use of OKDF2.

— They shall agree on a specific hash-function h, with output bit-length L_h, which shall be one of the algorithms specified in ISO/IEC 10118 (all parts).

— They shall agree on the method for binary encoding the counter c, including the number L_c of bits in the encoding. They shall also agree on the methods for encoding s, a, t (if used) and u (if used) prior to their concatenation in step d) 1) of 6.3.3.

— They shall agree on the algorithm and bit-length L_k for the output secret key k, together with an algorithm identifier a for this algorithm.

— If the option to employ a salt is exercised, they shall agree on a specific salt value t, which does not need to be a secret.

— They shall possess a bit string s, constituting the secret information, together (optionally) with an auxiliary secret u.

6.3.3 Operation of function

The function involves the following steps.

a) Set $d = \lceil L_k / L_h \rceil$.

b) If $d \geq 2^{L_c}$, then halt and output "invalid."

c) Set z = empty bit string.

d) For $c = 1$ to d:

   1) Set $z = z \,||\, h(s \,||\, a \,||\, c \,||\, [t] \,||\, [u])$.

e) Set k = the leftmost L_k bits of z.

6.4 One-step key derivation function 3 (OKDF3)

6.4.1 General

This key derivation function takes as input secret information s, (optionally) a salt t, which does not need to be secret, and (optionally) auxiliary secret information u. It gives as output a bit string b which can be partitioned into one or more secret keys. If the key derivation function is invoked multiple times with the same s to generate a series of distinct secret keys, then a different value of t or u shall be used with each invocation.

NOTE The "NIST/SP 800-56A concatenation key derivation function" mechanism and "NIST/SP 800-56A ASN.1 key derivation function" mechanism described in ISO/IEC 11770-3[3], as well as option 1 of the single-step KDF mechanism specified in NIST/SP 800-56A (Revision 2):2013, 5.8.1.1[9] and NIST/SP 800-56B (Revision 1):2014, 5.5.1.1[10] are special cases of the OKDF3 mechanism, where the counter c is encoded as a 32-bit, big-endian bit string, and t is the appropriate encoding of OtherInfo.

6.4.2 Requirements for use

Prior to use, the one or more entities which wish to use the function shall meet the following requirements.

— They shall agree on the use of OKDF3.

— They shall agree on a specific hash-function h, with output bit-length L_h, which shall be one of the algorithms specified in ISO/IEC 10118 (all parts).

— They shall agree on the method for binary encoding the counter c, including the number L_c of bits in the encoding. They shall also agree on the methods for encoding s, t (if used) and u (if used) prior to their concatenation in step d) 1) of 6.4.3.

— They shall agree on the length L_b of the required output string and precisely how the output b of the KDF shall be partitioned into secret keys.

— If the option to employ a salt is exercised, they shall agree on a specific salt value t, which does not need to be a secret.

— They shall possess a bit string s, constituting the secret information, together (optionally) with an auxiliary secret u.

6.4.3 Operation of function

The function involves the following steps.

a) Set $d = \lceil L_b / L_h \rceil$.

b) If $d \geq 2^{L_c}$, then halt and output "invalid."

c) Set z = empty bit string.

d) For $c = 1$ to d:

   1) Set $z = z \,||\, h(c \,||\, s \,||\, t \,||\, [u])$.

e) Set b = the leftmost L_b bits of z.

6.5 One-step key derivation function 4 (OKDF4)

6.5.1 General

This key derivation function takes as input secret information s, a label p, which describes the intended use of the output, (optionally) a salt t, which does not need to be a secret, and (optionally) auxiliary

secret information u. It gives as output a bit string b which can be partitioned into one or more secret keys. If the key derivation function is invoked multiple times with the same s to generate a series of distinct secret keys, then a different value of p, t, or u shall be used with each invocation.

NOTE Provided that $p \,||\, [t] \,||\, [u]$ is an appropriate encoding of SharedInfo, the "ANSI X9.63 key derivation function" mechanism described in ISO/IEC 11770-3[3] is a special case of the OKDF4 mechanism, where the counter c is encoded as a 32-bit, big-endian bit string.

6.5.2 Requirements for use

Prior to use, the one or more entities which wish to use the function shall meet the following requirements.

— They shall agree on the use of OKDF4.

— They shall agree on a specific hash-function h, with output bit-length L_h, which shall be one of the algorithms specified in ISO/IEC 10118 (all parts).

— They shall agree on the method for binary encoding the counter c, including the number L_c of bits in the encoding. They shall also agree on the methods for encoding s, p, t (if used) and u (if used) prior to concatenation in step d) 1) of 6.5.3.

— They shall agree on the length L_b of the required output string and precisely how the output b of the KDF shall be partitioned into secret keys. They shall also agree on a label p that specifies this partitioning and with which algorithm each of the keys shall be used.

— If the option to employ a salt is exercised, they shall agree on a specific salt value t, which does not need to be a secret.

— They shall possess a bit string s, constituting the secret information, together (optionally) with an auxiliary secret u.

6.5.3 Operation of function

The function involves the following two steps.

a) Set $d = \lceil L_b \,/\, L_h \rceil$.

b) If $d \geq 2^{L_c}$, then halt and output "invalid."

c) Set z = empty bit string.

d) For $c = 1$ to d:

   1) Set $z = z \,||\, h(s \,||\, c \,||\, p \,||\, t \,||\, [u])$.

e) Set b = the leftmost L_b bits of z.

6.6 One-step key derivation function 5 (OKDF5)

6.6.1 General

This key derivation function takes as input secret information s, (optionally) a salt t, which does not need to be secret, and (optionally) auxiliary secret information u. It gives as output a bit string b which can be partitioned into one or more secret keys. If the key derivation function is invoked multiple times

with the same s to generate a series of distinct secret keys, then a different value of t or u shall be used with each invocation.

NOTE The "KDF1" mechanism specified in ISO/IEC 18033-2[6] is a special case of the OKDF5 mechanism, where the counter c is encoded as a 32-bit octet string, the methods employed for encoding s, t (if used) and u (if used) produce an octet string as v, and $e = 0$. Similarly, the "KDF2" mechanism specified in ISO/IEC 18033-2[6] is a special case of the OKDF5 mechanism, where the counter c is encoded as a 32-bit octet string, the methods employed for encoding s, t (if used) and u (if used) produce an octet string as v, and $e = 1$.

6.6.2 Requirements for use

Prior to use, the one or more entities which wish to use the function shall meet the following requirements.

— They shall agree on the use of OKDF5.

— They shall agree on a specific hash-function h, with output bit-length L_h, which shall be one of the algorithms specified in ISO/IEC 10118 (all parts).

— They shall agree on the method for binary encoding the counter c, including the number L_c of bits in the encoding. They shall also agree on the methods for encoding s, t (if used) and u (if used) to create v in step d) 1) of 6.6.3 prior to concatenation in step d) 2) of 6.6.3.

— They shall agree on the length L_b of the required output string and precisely how the output b of the KDF shall be partitioned into secret keys.

— If the option to employ a salt is exercised, they shall agree on a specific salt value t, which does not need to be a secret.

— They shall possess a bit string s, constituting the shared secret information, together (optionally) with an auxiliary secret u.

— They shall agree on a start value e for the counter c, where e shall be 0 or 1.

6.6.3 Operation of function

The function involves the following steps.

a) Set $d = \lceil L_b / L_h \rceil$.

b) If $e+d-1 \geq 2^{L_c}$, then halt and output "invalid."

c) Set z = empty bit string.

d) For $c = e$ to $e+d-1$:

   1) Let v be an agreed encoding of s, t and (optionally) u.

   2) Set $z = z \,||\, h(v \,||\, c)$.

e) Set b = the leftmost L_b bits of z.

6.7 One-step key derivation function 6 (OKDF6)

6.7.1 General

This key derivation function takes as input secret information s, a salt t', which does not need to be a secret, but shall take the form of an appropriate MAC key, (optionally) a salt t, which does not need to be secret, and (optionally) auxiliary secret information u. It gives as output a bit string b which can be partitioned into one or more secret keys. If the key derivation function is invoked multiple times with

the same s to generate a series of distinct secret keys, then a different value of t', t or u shall be used with each invocation.

NOTE Option 2 of the single-step KDF mechanism specified in NIST/SP 800-56A (Revision 2):2013, 5.8.1.1[9], and NIST/SP 800-56B (Revision 1):2014, 5.5.1.1[10] is a special case of the OKDF6 mechanism, where the counter c is encoded as a 32-bit, big-endian bit string, and t is the appropriate encoding of OtherInfo.

6.7.2 Requirements for use

Prior to use, the one or more entities which wish to use the function shall meet the following requirements.

— They shall agree on the use of OKDF6.

— They shall agree on a specific MAC algorithm f, with output bit-length L_f, which shall be one of the algorithms specified in ISO/IEC 9797 (all parts).

— They shall agree on the method for binary encoding the counter c, including the number L_c of bits in the encoding. They shall also agree on the methods for encoding s, t (if used) and u (if used) prior to concatenation in step d) 1) of 6.7.3.

— They shall agree on the length L_b of the required output string b, and precisely how the output b of the KDF shall be partitioned into secret keys.

— If the option to employ the salt t is exercised, they shall agree on a specific value for t, which does not need to be a secret.

— They shall agree on a specific value for the salt t', which shall take the form of a MAC key for the chosen MAC algorithm f. The value of t' does not need to be a secret.

— They shall share a bit string s, constituting the secret information, together (optionally) with an auxiliary secret u.

6.7.3 Operation of function

The function involves the following steps.

a) Set $d = \lceil L_b / L_f \rceil$.

b) If $d \geq 2^{L_c}$, then halt and output "invalid."

c) Set z = empty bit string.

d) For $c = 1$ to d:

   1) Set $z = z \,||\, f_{t'}(c \,||\, s \,||\, t \,||\, [u])$.

e) Set b = the leftmost L_b bits of z.

7 Two-step key derivation functions

7.1 General

A two-step KDF (a TKDF) is a KDF which transforms input information into one or more keys in two operations. Like all KDFs, a TKDF takes as input secret information, together with other parameters, some or all of which might be public and gives as output one or more secret keys.

A TKDF consists of a combination of two functions: a key extraction function and a key expansion function. The key extraction function and key expansion functions specified in this document are all

based on MAC algorithms. The KTF is executed to produce a MAC key; this MAC key is then used as one of the inputs to the KPF.

If a MAC key is already available for use by the KPF, then the key extraction step is unnecessary, i.e. the TKDF simply involves applying a KPF to the input, in which case the secret information input to the TKDF shall take the form of a MAC key suitable for use with the MAC algorithm on which the KPF is based. Under such circumstances, the TKDF is essentially one of the key expansion functions specified in 7.3. This key expansion function shall satisfy the KDF requirements specified in 5.1, with the MAC key input to the KPF serving as the secret *s*.

In this document, one (non-identity) key extraction function (KTF1), four key expansion functions (KPF1 to KPF4), and four two-step KDFs (TKDF1 to TKDF4) are specified. In general, KTF1 can be used with any of KPF1 to KPF4 to create a corresponding TKDF.

In KTF1, a MAC function f is used with a MAC key that may be publicly known, but with secret information as the input message data. The MAC algorithm employed by the KTF1 mechanism shall satisfy the following property, in addition to those that are ordinarily required (cf. 3.7).

Given that the key k input to the MAC function f is a publicly known value and the input message data consists of secret information s, then, without prior knowledge of s, it shall be computationally infeasible to predict the (as-yet-unseen) MAC value $f_k(s)$ with a probability of success that is a significant improvement over simply guessing either $f_k(s)$ or s. This shall be the case even when given knowledge of a (bounded) number of MAC values $f_{kj}(s)$, computed using the same (unknown) s with MAC keys k_j that are known and (adaptively) chosen, subject to the restriction that k_j is not equal to k.

NOTE If a KTF and a KPF are used together to form a TKDF, then the MAC algorithms on which the KTF and KPF are based need not be the same. However, there might be implementation advantages deriving from use of the same MAC algorithm.

7.2 Key extraction function

7.2.1 Key extraction function 1 (KTF1)

7.2.1.1 General

This key extraction function takes as input a salt t, which does not need to be secret, and secret information s. It gives as output a MAC key k_m.

NOTE If used with the HMAC MAC algorithm, as specified in ISO/IEC 9797-2, then KTF1 is equal to the KTF specified by Krawczyk[7] [8], which forms part of the key derivation function known as HKDF. When an HMAC MAC algorithm is employed, an instantiation of the "randomness extraction" mechanism specified in NIST/SP 800-56C:2011, Section 5[11] is a special case of the KTF1 mechanism, in which no truncation is employed (i.e. $L_f = L_k$).

7.2.1.2 Requirements for use

Prior to use, the one or more entities which wish to use the function shall meet the following requirements.

— They shall agree on the use of KTF1.

— They shall agree on a target key length L_k, i.e. the key length of the MAC function with which the output k_m of the KTF is to be used.

— They shall agree on a specific MAC function f, which shall be one of the algorithms specified in ISO/IEC 9797 (all parts). This MAC function shall give as output a MAC of bit-length greater than or equal to the target key length L_k, i.e. $L_f \geq L_k$.

— They shall agree on the method for binary encoding s prior to application of the MAC function in step a) of 7.2.1.3.

— They shall agree on a specific salt value t, which shall take the form of a MAC key for the chosen MAC algorithm. The salt does not need to be a secret.

— They shall possess a bit string s, constituting the secret information.

7.2.1.3 Operation of function

The function involves the following two steps.

a) Compute $f_t(s)$.

b) Set k_m to be the leftmost L_k bits of $f_t(s)$.

7.3 Key expansion functions

7.3.1 Key expansion function 1 (KPF1)

7.3.1.1 General

This key expansion function takes as input a salt t, which does not need to be secret, and a MAC key k_m. It gives as output a bit string b which can be partitioned into one or more secret keys. If the key expansion function is invoked multiple times with the same k_m to generate a series of distinct secret keys, then a different value of t shall be used with each invocation.

NOTE If used with an HMAC MAC algorithm f as specified in ISO/IEC 9797-2, with a MAC key k_m consisting of an octet string of bit-length at least L_h (the bit-length of the octet strings output by the hash-function h used by f), with a counter c encoded as a single 8-bit octet, and with an output b consisting of an octet string of bit-length no greater than 255 L_h, then such an instantiation of KPF1 is consistent with the definition of HKDF-Expand, as specified by Krawczyk[8], which forms part of the key derivation function known as "HKDF."

7.3.1.2 Requirements for use

Prior to use, the one or more entities which wish to use the function shall meet the following requirements.

— They shall agree on the use of KPF1.

— They shall agree on a specific MAC algorithm f, with output bit-length L_f, which shall be one of the algorithms specified in ISO/IEC 9797 (all parts).

— They shall agree on the method for binary encoding the counter c, including the number L_c of bits in the encoding. They shall also agree on the methods for encoding y and t prior to concatenation in step e) 1) of 7.3.1.3.

— They shall agree on the length L_b of the required output string b, and precisely how the output b of the KDF shall be partitioned into secret keys.

— They shall agree on a specific salt value t. The salt does not need to be a secret.

— They shall possess a secret MAC key k_m.

7.3.1.3 Operation of function

The function involves the following steps.

a) Set $d = \lceil L_b / L_f \rceil$.

b) If $d \geq 2^{L_c}$, then halt and output "invalid."

c) Set y = empty bit string.

d) Set $z = y$.

e) For $c = 1$ to d:

1) Set $y = f_{km}(y \,||\, t \,||\, c)$.

2) Set $z = z \,||\, y$.

f) Set b = the leftmost L_b bits of z.

7.3.2 Key expansion function 2 (KPF2)

7.3.2.1 General

This key expansion function takes as input a salt t, which does not need to be secret, a label p, which describes the intended use of the output, and a MAC key k_m. It gives as output a bit string b which can be partitioned into one or more secret keys. If the key expansion function is invoked multiple times with the same k_m to generate a series of distinct secret keys, then a different value of t or p shall be used with each invocation.

NOTE If the "KDF in Counter Mode" mechanism specified in NIST/SP 800-108:2009, 5.1[12] is implemented with the suggested ordering of the data fields constituting the "messages" that are input to the MAC algorithm (including the counter in the leftmost position and the intended bit-length of the output in the rightmost position), then that instance of the "KDF in Counter Mode" mechanism is a special case of the KPF2 mechanism.

7.3.2.2 Requirements for use

Prior to use, the one or more entities which wish to use the function shall meet the following requirements.

— They shall agree on the use of KPF2.

— They shall agree on a specific MAC algorithm f, with output bit-length L_f, which shall be one of the algorithms specified in ISO/IEC 9797 (all parts).

— They shall agree on the method for binary encoding the counter c, including the number L_c of bits in the encoding. They shall also agree on the method of encoding of p, t and L_b prior to concatenation in step d) 1) of 7.3.2.3.

— They shall agree on the length L_b of the required output string b, and precisely how the output b of the KDF shall be partitioned into secret keys. They shall also agree on a label p that specifies this partitioning and with which algorithm each of the keys shall be used.

— They shall agree on a specific salt value t. The salt does not need to be a secret.

— They shall possess a secret MAC key k_m.

7.3.2.3 Operation of function

The function involves the following steps.

a) Set $d = \lceil L_b / L_f \rceil$.

b) If $d \geq 2^{L_c}$, then halt and output "invalid."

c) Set z = empty bit string.

d) For $c = 1$ to d:

1) Set $z = z \,||\, f_{km}(c \,||\, p \,||\, t \,||\, L_b)$.

e) Set b = the leftmost L_b bits of z.

7.3.3 Key expansion function 3 (KPF3)

7.3.3.1 General

This key expansion function takes as input a pair of salts t and t', which do not need to be a secret, a label p, which describes the intended use of the output, and a MAC key k_m. It gives as output a bit string b which can be partitioned into one or more secret keys. If the key expansion function is invoked multiple times with the same k_m to generate a series of distinct secret keys, then a different value of t, t' or p shall be used with each invocation.

NOTE If an instantiation of the "KDF in Feedback Mode" mechanism specified in NIST/SP 800-108:2009, 5.2[12] is implemented with the suggested ordering of the data fields constituting the "messages" that are input to the MAC algorithm — including the counter (when used) in the second leftmost position and the intended bit-length of the output in the rightmost position — then that instance of the "KDF in Feedback Mode" mechanism is a special case of the KPF3 mechanism.

7.3.3.2 Requirements for use

Prior to use, the one or more entities which wish to use the function shall meet the following requirements.

— They shall agree on the use of KPF3.

— They shall agree on a specific MAC algorithm f, with output bit-length L_f, which shall be one of the algorithms specified in ISO/IEC 9797 (all parts).

— They shall agree on an integer value for M_c, which specifies the maximum value permitted for the iteration counter c, and thus determines the maximum number of times that the MAC algorithm f can be invoked during a single execution of KPF3.

— They shall agree on whether or not the counter value c is to be used as part of the message data input to the MAC algorithm. If the counter value is to be used as input to the MAC algorithm, then they shall agree on the method used for the binary encoding of c, including the number L_c of bits in the encoding; in this case, M_c shall be less than 2^{Lc}.

— They shall agree on the methods of encoding of y (including its initial value, t'), p, t and L_b prior to their use in the assignment in step c) and the concatenations in step e) of 7.3.3.3.

— They shall agree on the length L_b of the required output string b, and precisely how the output b of the KDF shall be partitioned into secret keys. They shall also agree on a label p that specifies this partitioning and with which algorithm each of the keys shall be used.

— They shall agree on two specific salt values t and t'. The salt values do not need to be secret.

— They shall possess a secret MAC key k_m.

7.3.3.3 Operation of function

The function involves the following steps (if a counter value c is in use).

a) Set $d = \lceil L_b / L_f \rceil$.

b) If $d > M_c$, then halt and output "invalid."

c) Set $y = t'$.

d) Set z = empty string.

e) For $c = 1$ to d:

1) Set $y = f_{km}(y \,||\, [c] \,||\, p \,||\, t \,||\, L_b)$.

2) Set $z = z \,||\, y$.

f) Set b = the leftmost L_b bits of z.

NOTE When deciding whether or not to include c as part of the input to the MAC algorithm, one should consider the possibility of encountering fixed points or short cycles in the values of y produced by iterating the computation $y = f_{km}(y \,||\, p \,||\, t \,||\, L_b)$.

7.3.4 Key expansion function 4 (KPF4)

7.3.4.1 General

This key expansion function takes as input a salt t, which does not need to be secret, a label p, which describes the intended use of the output, and a MAC key k_m. It gives as output a bit string b which can be partitioned into one or more secret keys. If the key expansion function is invoked multiple times with the same k_m to generate a series of distinct secret keys, then a different value of p or t shall be used with each invocation.

NOTE If an instantiation of the "KDF in Double-Pipeline Mode" mechanism specified in NIST/SP 800-108:2009, 5.3[12] is implemented with the suggested ordering of the data fields constituting the "messages" that are input to the MAC algorithm — including the counter (when used) in the second leftmost position and the intended bit-length of the output in the rightmost position — then that instance of the "KDF in Double-Pipeline Mode" mechanism is a special case of the KPF4 mechanism.

7.3.4.2 Requirements for use

Prior to use, the one or more entities which wish to use the function shall meet the following requirements.

— They shall agree on the use of KPF4.

— They shall agree on a specific MAC algorithm f, with output bit-length L_f, which shall be one of the algorithms specified in ISO/IEC 9797 (all parts).

— They shall agree on an integer value for M_c, which specifies the maximum value permitted for the iteration counter c, and thus determines the maximum number of times ($2M_c$) that the MAC algorithm f can be invoked during a single execution of KPF4.

— They shall agree on whether or not the value of the iteration counter c is to be used as part of the message data input to the MAC algorithm. If the counter value is to be used as input to the MAC algorithm, then they shall agree on the method used for the binary encoding of c, including the number L_c of bits in the encoding; in this case, M_c shall be less than 2^{Lc}.

— They shall agree on the methods of encoding of y, p, t and L_b prior to their use in the concatenations in steps c) and e) of 7.3.4.3.

— They shall agree on the length L_b of the required output string b and precisely how the output b of the KDF shall be partitioned into secret keys. They shall also agree on a label p that specifies this partitioning and with which algorithm each of the keys shall be used.

— They shall agree on a specific salt value t. The salt value does not need to be secret.

— They shall possess a secret MAC key k_m.

7.3.4.3 Operation of function

The function involves the following steps (if a counter value c is in use).

a) Set $d = \lceil L_b / L_f \rceil$.

b) If $d > M_c$, then halt and output "invalid."

c) Set $y = p \,||\, t \,||\, L_b$.

d) Set z = empty string.

e) For $c = 1$ to d:

   1) Set $y = f_{km}(y)$.

   2) Set $z = z \,||\, f_{km}(y \,||\, [c] \,||\, p \,||\, t \,||\, L_b)$.

f) Set b = the leftmost L_b bits of z.

NOTE When deciding whether or not to include c as part of the input to the MAC algorithm, one should consider the possibility of encountering fixed points or short cycles in the values of y produced by iterating the computation $y = f_{km}(y)$, which would cause repetitions in the value of $f_{km}(y \,||\, p \,||\, t \,||\, L_b)$.

7.4 Two-step KDFs

7.4.1 Two-step key derivation function 1 (TKDF1)

7.4.1.1 General

This key derivation function takes as input a pair of salt values, one for the KTF and one for the KPF, which do not need to be secret, and secret information s. It gives as output a bit-string b which can be partitioned into one or more secret keys. If the key derivation function is invoked multiple times with the same s to generate a series of secret keys, then in each invocation, a different value shall be used for at least one of the salt values.

7.4.1.2 Requirements for use

The requirements for use of this mechanism are those given in 7.2.1.2 and 7.3.1.2.

7.4.1.3 Operation of function

The function involves the following two steps.

a) Use KTF1 with s as input to extract a MAC key k_m.

b) Use KPF1 with k_m as input to generate the bit-string b.

7.4.2 Two-step key derivation function 2 (TKDF2)

7.4.2.1 General

This key derivation function takes as input a pair of salt values, one for the KTF and one for the KPF, which do not need to be secret, a label p, which describes the intended use of the output, and secret information s. It gives as output a bit string b which can be partitioned into one or more secret keys. If the key derivation function is invoked multiple times with the same s to generate a series of distinct secret keys, then in each invocation a different value shall be used either for the label or for at least one of the salt values.

7.4.2.2 Requirements for use

The requirements for use of this mechanism are those given in 7.2.1.2 and 7.3.2.2.

