INTERNATIONAL STANDARD

ISO/IEC 10118-3

Fourth edition
2018-10

IT Security techniques — Hash-functions —

Part 3: **Dedicated hash-functions**

Techniques de sécurité IT — Fonctions de brouillage —

Partie 3: Fonctions de brouillage dédiées

Reference number
ISO/IEC 10118-3:2018(E)

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.

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) or the IEC list of patent declarations received (see http://patents.iec.ch).

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

For an explanation of the voluntary nature of standards, 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 www.iso.org/iso/foreword.html.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

This fourth edition cancels and replaces the third edition (ISO/IEC 10118-3:2004), which has been technically revised. It also incorporates the Amendment ISO/IEC 10118-3:2004/Amd1:2006 and Technical Corrigendum ISO/IEC 10118-3:2004/Cor1:2011.

The main changes compared to the previous edition are as follows:

— SHA-3, STREEBOG and SM3 hash-functions have been included;

— SHA-3 extendable-output functions have been included;

— cautionary notes for hash-functions with short hash-codes have been added.

A list of all parts in the ISO/IEC 10118 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

IT Security techniques — Hash-functions —

Part 3: Dedicated hash-functions

1 Scope

This document specifies dedicated hash-functions, i.e. specially designed hash-functions. The hash-functions in this document are based on the iterative use of a round-function. Distinct round-functions are specified, giving rise to distinct dedicated hash-functions.

The use of Dedicated Hash-Functions 1, 2 and 3 in new digital signature implementations is deprecated.

NOTE As a result of their short hash-code length and/or cryptanalytic results, Dedicated Hash-Functions 1, 2 and 3 do not provide a sufficient level of collision resistance for future digital signature applications and they are therefore, only usable for legacy applications. However, for applications where collision resistance is not required, such as in hash-functions as specified in ISO/IEC 9797-2, or in key derivation functions specified in ISO/IEC 11770-6, their use is not deprecated.

Numerical examples for dedicated hash-functions specified in this document are given in Annex B as additional information. For information purposes, SHA-3 extendable-output functions are specified in Annex C.

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 10118-1, *Information technology — Security techniques — Hash-functions — Part 1: General*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 10118-1 and the following apply.

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

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

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

3.1
block
bit string of length L_1, i.e., the length of the first input to the round-function

3.2
word
string of bits

3.3
circulant matrix
matrix with the property that each row, apart from the first, consists of the right cyclic shift by one position of the row immediately above it

3.4
abelian group
group (G, *) such that $a*b = b*a$ for every a and b in G

3.5
field
set of elements S and a pair of operations (+,*) defined on S such that: (i) $a*(b + c) = a*b + a*c$ for every a, b and c in S, (ii) S together with + forms an abelian group (with identity element 0) and (iii) S excluding 0 together with * forms an abelian group

4 Symbols

4.1 Symbols specified in ISO/IEC 10118-1

B_i	byte
D	data
H	hash-code
IV	initializing value
L_1	length (in bits) of the first of the two input strings to the round-function Φ
L_2	length (in bits) of the second of the two input strings to the round-function Φ, of the output string from the round-function Φ and of the IV
L_X	length (in bits) of a bit string X
$X \oplus Y$	bitwise exclusive-or of bit strings X and Y (where $L_X = L_Y$)
$X \mid\mid Y$	concatenation of strings of bits X and Y in the indicated order
Φ	a round-function, i.e. if X, Y are bit strings of lengths L_1 and L_2 respectively, then $\Phi(X, Y)$ is the string obtained by applying Φ to X and Y

4.2 Symbols specific to this document

A^i	sequence of constant matrices used in the specification of the round-function defined in Clause 16
A^n	concatenation of n instances of the word A
a_i, a'_i	sequences of indices used in specifications of a round-function
C_i, C'_i	constant words used in the round-functions
C''	8 × 8 circulant matrix with entries chosen from GF(2^8) used in the specification of the round-function in Clause 16
c_0	function taking a string of 64 elements of GF(2^8) as input and giving an 8 × 8 matrix with entries from GF(2^8) as output, used in specifying the round-function defined in Clause 16
c_1, c_2, c_3	functions taking an 8 × 8 matrix of elements of GF(2^8) as input and giving an 8 × 8 matrix with entries from GF(2^8) as output, used in the specification of the round-function defined in Clause 16

c_4	function taking two 8 × 8 matrices of elements of GF(2^8) as input and giving an 8 × 8 matrix with entries from GF(2^8) as output, used in the specification of the round-function defined in Clause 16
D_i	a block derived from the data string after the padding process
d_i, e_i, f_i, g_i	functions taking either one or three words as input and producing a single word as output, used in specifying round-functions
H_i	a string of L_2 bits which is used in the hashing operation to store an intermediate result
Int_n	an inverse mapping to the mapping Vec_n, i.e. $\mathrm{Int}_n = \mathrm{Vec}_n^{-1}$
GF(2^8)	a field defined as GF(2)[x] / $p_8(x)$ where $p_8(x) = x^8 + x^4 + x^3 + x^2 + 1$. The elements of the field are 8-bit strings
M	an 8 × 8 matrix whose entries are chosen from GF(2^8)
q	number of blocks in the data string after the padding and splitting processes
$R^n()$	operation of right shift by n bits, i.e. if A is a word and n is a non-negative integer then $R^n(A)$ denotes the word obtained by right-shifting the contents of A by n positions
$S^n()$	operation of “circular left shift” by n bit positions, i.e. if A is a word and n is a non-negative integer then $S^n(A)$ denotes the word obtained by left-shifting the contents of A by n places in a cyclic fashion
$S'^n()$	operation of “circular right shift” by n bit positions, i.e. if A is a word and n is a non-negative integer then $S'^n(A)$ denotes the word obtained by right-shifting the contents of A by n places in a cyclic fashion
s	a function, which replaces an element $x \in$ GF(2^8) with another element $s[x] \in$ GF(2^8)
t_i, t'_i	shift-values used in specifying a round-function
Vec_n	a bijective mapping from $\mathbb{Z}_{2^n}$ to the set of n-bit words, which maps an integer from $\mathbb{Z}_{2^n}$ to its binary representation (i.e. for any integer $z = z_0 + 2z_1 + ... + 2^{n-1}z_{n-1}$ of the set $\mathbb{Z}_{2^n}$, where $z_j \in \{0,1\}$, $j = 0,..., n - 1$, by definition $\mathrm{Vec}_n(z) = (z_{n-1}\|...\|z_1\|z_0)$)
W, X_i, X'_i, Y_i, Z_i	words used to store the results of intermediate computations
W', X'', K_i, Y', Z'	matrices with entries chosen from GF(2^8) used to store the results of intermediate computations
$\mathbb{Z}_{2^n}$	set of non-negative integers less than 2^n, together with the operations of addition and multiplication modulo 2^n
$\wedge$	bitwise logical AND operation on bit strings, i.e. if A, B are words then $A \wedge B$ is the word equal to bitwise logical AND of A and B
$\vee$	bitwise logical OR operation on bit strings, i.e. if A, B are words then $A \vee B$ is the word equal to bitwise logical OR of A and B
$\neg$	bitwise logical NOT operation on a bit string, i.e. if A is a word then $\neg A$ is the word equal to the bitwise logical NOT of A

$\uplus$	addition modulo 2^w operation, where w is the number of bits in a word; i.e. if A and B are w–bit words, then A $\uplus$ B is the word obtained by treating A and B as the binary representations of integers and computing their sum modulo 2^w, where the result is constrained to lie between 0 and $2^w - 1$ inclusive. The value of w is 32 for Dedicated Hash-Functions 1 to 4, defined in Clauses 7 to 10, 64 for Dedicated Hash-Functions 5 and 6, defined in Clauses 11 and 12 and 512 for Dedicated Hash-Functions 11 and 12, defined in Clauses 17 and 18
$\cdot$	multiplication operation of 8 × 8 matrices with entries chosen from GF(2^8); i.e. if A and B are such matrices, then A·B is the matrix obtained by multiplying A and B in the following way. Treat each entry of either A or B as the binary polynomial representation of an integer (for example, the binary polynomial representation of integer `89` (hexadecimal) is $x^7 + x^3 + 1$); treat the multiplication of two of the entries as the remainder when multiplication of the two polynomials is divided by a polynomial $p_8(x)$, where $p_8(x) = x^8 + x^4 + x^3 + x^2 + 1$; and treat the sum operation as the operation $\oplus$
:=	a symbol denoting the "set equal to" operation used in procedural specifications of round-functions, where it indicates that the value of the variable (e.g. word or matrix) on the left side of the symbol should be set equal to the value of the expression on the right side of the symbol

5 Requirements

Users who wish to employ a hash-function from this document shall select

— one of the dedicated hash-functions specified below, and

— the length, L_H, of the hash-code H.

NOTE 1 All the hash-functions defined in this document take a bit string as input and give a bit string as output; this is independent of the internal byte ordering convention used within each hash-function.

NOTE 2 The choice of L_H affects the security of the hash-function. All of the hash-functions specified in this document are believed to be collision-resistant hash-functions in environments where performing $2^{L_H/2}$ hash-code computation is deemed to be computationally infeasible.

Annex A defines object identifiers that shall be used to identify the hash-functions specified in this document.

6 Models for dedicated hash-functions

6.1 Use of models

The 17 dedicated hash-functions specified in this document are defined using two different models. Dedicated Hash-Functions 1 to 12 and 17 are defined using the general round-function-based model defined in ISO/IEC 10118-1, which is further described in 6.2. Dedicated Hash-Functions 13 to 16 use the sponge construction model as defined in 6.3.

6.2 Round-function model

Dedicated Hash-Functions 1 to 12 and 17 specified in this document are based on the general model for hash-functions given in ISO/IEC 10118-1.

In the specifications of the hash-functions in this document, it is assumed that the padded data string input to the hash-function is in the form of a sequence of bytes. If the padded data string is in the form of a sequence of $8n$ bits, $x_0, x_1, ..., x_{8n-1}$, then it shall be interpreted as a sequence of n bytes, $B_0, B_1, ..., B_{n-1}$, in the following way. Each group of eight consecutive bits is considered as a byte, the first bit of a group being the most significant bit of that byte. Hence,

$$B_i = 2^7 x_{8i} + 2^6 x_{8i+1} + \dots + x_{8i+7}$$

for every i $(0 \le i < n)$.

The output transformation for the hash-functions specified in this document is defined so that the hash-code, H, is derived by taking the left-most L_H bits of the final L_2-bit output string H_q.

Identifiers are defined for each of the 17 dedicated hash-functions specified in this document. The hash-function identifiers for the dedicated hash-functions specified in Clauses 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22 and 23 are equal to `31`, `32`, `33`, `34`, `35`, `36`, `37`, `38`, `39`, `3A`, `3B`, `3C`, `3D`, `3E`, `3F`, `40` and `11` (hexadecimal), respectively. Some hash-function identifiers are also used in the OSI object identifiers assigned in Annex A.

6.3 Sponge model

In 6.3, a permutation-based hash-function with sponge construction is specified.

A permutation-based hash-function with sponge construction is defined by a padding method, a permutation and a set of parameters.

The sponge construction[4] is a framework for specifying functions on a binary data with arbitrary output length. The construction employs the following three components:

— an underlying function on fixed-length strings, denoted by f;

— a parameter called the rate, denoted by r;

— a padding rule, denoted by pad.

The function that the construction produces from these components is called a sponge function, denoted by SPONGE[f, pad, r]. A sponge function takes two inputs, a bit string, N, and the bit length, d, of the output string, SPONGE[f, pad, r](N, d).

NOTE For further details on the rationale of the sponge construction framework, see Reference [5].

The function, f, maps strings of a single, fixed length, b, to strings of the same length. b is called the width of f. When the underlying function, f, is invertible, i.e. a permutation, it is a permutation-based hash-function with sponge construction.

The rate, r, is a positive integer that is strictly less than the width b. The capacity, c, is the positive integer $b - r$. Thus, $r + c = b$.

In the padding rule, pad is a function that produces padding, i.e. a string with an appropriate length to append to another string. In general, given a positive integer x and a non-negative integer m, the output pad(x, m) is a string with the property that m + len[pad(x, m)] is a positive multiple of x. Within the sponge construction, $x = r$ and m = len(N), so that the padded input string can be partitioned into a sequence of r-bit strings.

Given these three components, f, *pad* and r, as described above, the SPONGE[f, pad, r] function on (N, d) is specified by SPONGE[f, pad, r](N, d). The width b is determined by the choice of f.

SPONGE[f, pad, r](N, d)

Input: string N, non-negative integer d

Output: string Z, such that len(Z) = d

Steps:

a) Let $P = N$ || pad[r, len(N)].

b) Let q = len(P)/r.

c) Let $c = b - r$.

d) Let $P_0, \ldots, P_{q-1}$ be the unique sequence of strings of length r, such that $P = P_0 \,||\, \ldots \,||\, P_{q-1}$.

e) Let $S = 0^b$.

f) For i from 0 to $q-1$, let $S = f\,[S \oplus (P_i \,||\, 0^c)]$.

g) Let Z be the empty string.

h) Let $Z = Z \,||\, Trunc_r(S)$.

i) If $d \leq |Z|$, then return $Trunc_d(Z)$; else, continue.

j) Let $S = f(S)$ and continue with step h).

Note that the input d determines the number of bits that SPONGE[f, pad, r](N, d) returns, but it does not affect their values. In principle, the output can be regarded as an infinite string, whose computation, in practice, is halted after the desired number of output bits is produced.

The parameters of a sponge construction include:

— b, the width;

— r, the rate;

— c, the capacity, such that $b = r + c$; and

— d, the output length.

Here, if notations specified in ISO/IEC 10118-1:2016, Clause 3 are used, r can be considered as L_1, which is the length of a block of input data (message), while b can be considered as L_2, which is the output length of the function f. Furthermore, the relation between function f and the round-function Φ as defined in ISO/IEC 10118-1:2016, Clause 3 can be represented as $\Phi\,(P_i, S_{i-1}) = f\,[S_{i-1} \oplus (P_i \,||\, 0^c)]$, where $P_i = D_i$ is the ith data block, while $S_{i-1} = H_{i-1}$ is the output of the previous execution. The squeezing stage is considered the output transformation.

7 Dedicated Hash-Function 1 (RIPEMD-160)

7.1 General

In Clause 7, a padding method, an initializing value and a round-function for use in the general model for hash-functions described in ISO/IEC 10118-1 are specified. The padding method, initializing value and round-function specified here, when used in the above general model, together define Dedicated Hash-Function 1. This dedicated hash-function can be applied to all data strings, D, containing at most $2^{64}-1$ bits.

The ISO/IEC hash-function identifier for Dedicated Hash-Function 1 is equal to `31` (hexadecimal).

NOTE 1 Dedicated Hash-Function 1 defined in Clause 7 is commonly called RIPEMD-160[6].

NOTE 2 As a result of a short hash-code length and/or cryptanalytic results, Dedicated Hash-Function 1 does not provide a sufficient level of collision resistance for future digital signature applications and it is, therefore, only used for legacy applications. However, for applications where collision resistance is not required, such as in hash-functions as specified in the ISO/IEC 9797 series or in key derivation functions specified in ISO/IEC 11770-6, its use is not deprecated.

7.2 Parameters, functions and constants

7.2.1 Parameters

For this hash-function, $L_1 = 512$, $L_2 = 160$ and L_H is up to 160.

7.2.2 Byte ordering convention

In the specification of the round-function of Clause 7, it is assumed that the block input to the round-function is in the form of a sequence of 32-bit words, each 512-bit block being made up of 16 such words. A sequence of 64 bytes, $B_0, B_1, ..., B_{63}$, shall be interpreted as a sequence of 16 words, $Z_0, Z_1, ..., Z_{15}$, in the following way. Each group of four consecutive bytes is considered as a word, the first byte of a word being the least significant byte of that word. Hence,

$$Z_i = 2^{24}B_{4i+3} + 2^{16}B_{4i+2} + 2^8 B_{4i+1} + B_{4i}, \quad (0 \le i \le 15).$$

To convert the hash-code from a sequence of words to a byte-sequence, the inverse process shall be followed.

NOTE The byte-ordering specified here is different from that of 9.2.2.

7.2.3 Functions

To facilitate software implementation, the round-function Φ is described in terms of operations on 32-bit words. A sequence of functions $g_0, g_1, ..., g_{79}$ is used in this round-function, where each function g_i, $0 \le i \le 79$, takes three words, X_0, X_1 and X_2, as input and produces a single word as output.

The functions g_i are defined as follows:

$$\begin{aligned}
g_i(X_0,X_1,X_2) &= X_0 \oplus X_1 \oplus X_2, & (0 \le i \le 15);\\
g_i(X_0,X_1,X_2) &= (X_0 \wedge X_1) \vee (\neg X_0 \wedge X_2), & (16 \le i \le 31);\\
g_i(X_0,X_1,X_2) &= (X_0 \vee \neg X_1) \oplus X_2, & (32 \le i \le 47);\\
g_i(X_0,X_1,X_2) &= (X_0 \wedge X_2) \vee (X_1 \wedge \neg X_2), & (48 \le i \le 63);\\
g_i(X_0,X_1,X_2) &= X_0 \oplus (X_1 \vee \neg X_2), & (64 \le i \le 79).
\end{aligned}$$

7.2.4 Constants

Two sequences of constant words, $C_0, C_1, ..., C_{79}$ and $C'_0, C'_1, ..., C'_{79}$, are used in this round-function. In a hexadecimal representation (where the most significant bit corresponds to the left-most bit), these are defined as follows:

C_i = 00000000,	$(0 \le i \le 15)$;
C_i = 5A827999,	$(16 \le i \le 31)$;
C_i = 6ED9EBA1,	$(32 \le i \le 47)$;
C_i = 8F1BBCDC,	$(48 \le i \le 63)$;
C_i = A953FD4E,	$(64 \le i \le 79)$
C'_i = 50A28BE6,	$(0 \le i \le 15)$;
C'_i = 5C4DD124,	$(16 \le i \le 31)$;
C'_i = 6D703EF3,	$(32 \le i \le 47)$;
C'_i = 7A6D76E9,	$(48 \le i \le 63)$;
C'_i = 00000000,	$(64 \le i \le 79)$.

Two sequences of 80 shift-values are used in this round-function, where each shift-value is between 5 and 15. These sequences are denoted by $(t_0, t_1, ..., t_{79})$ and $(t'_0, t'_1, ..., t'_{79})$. Two additional sequences of 80 indices are used in this round-function, where each value in the sequence is between 0 and 15. These sequences are denoted as $(a_0, a_1, ..., a_{79})$ and $(a'_0, a'_1, ..., a'_{79})$. All four sequences are defined in Table 1.