INTERNATIONAL STANDARD

ISO/IEC 10118-3

Second edition
2003-05-01

Information technology — Security techniques — Hash-functions —

Part 3: Dedicated hash-functions

Technologies de l'information — Techniques de sécurité — Fonctions de brouillage —

Partie 3: Fonctions de brouillage dédiées

Reference number
ISO/IEC 10118-3:2003(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

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 o 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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft Internationa Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication a an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of paten rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 10118-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology* Subcommittee SC 27, *IT Security techniques*.

This second edition cancels and replaces the first edition (ISO/IEC 10118-3:1998), which has been technicall revised.

ISO/IEC 10118 consists of the following parts, under the general title *Information technology — Securi techniques — Hash-functions*:

— *Part 1: General*

— *Part 2: Hash-functions using an n-bit block cipher*

— *Part 3: Dedicated hash-functions*

— *Part 4: Hash-functions using modular arithmetic*

Further parts may follow.

Information technology — Security techniques — Hash-functions —

Part 3: **Dedicated hash-functions**

1 Scope

This part of ISO/IEC 10118 specifies dedicated hash-functions, i.e., specially designed hash-functions. The hash-functions in this part of ISO/IEC 10118 are based on the iterative use of a round-function. Seven distinct round-functions are specified, giving rise to distinct dedicated hash-functions.

The first and third dedicated hash-functions in clauses 7 and 9 respectively provide hash-codes of lengths up to 160 bits; the second in clause 8 provides hash-codes of lengths up to 128 bits; the fourth in clause 10 provides hash-codes of lengths up to 256 bits; the sixth in clause 12 provides hash-codes of a fixed length, 384 bits; and the fifth and seventh in clauses 11 and 13 respectively provide hash-codes of lengths up to 512 bits.

2 Normative references

The following referenced documents are indispensable for the application 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: 2000, *Information technology — Security techniques — Hash-functions — Part 1: General*

3 Terms and definitions

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

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

3.2
word
a string of 32 bits used in dedicated hash-functions 1, 2, 3 and 4 of clauses 7, 8, 9 and 10 respectively, or a string of 64 bits used in dedicated hash-functions 5 and 6 of clauses 11 and 12 respectively.

3.3
matrix
an 8 by 8 matrix in which each entry is a string of 8 bits used in dedicated hash-function 7 of clause 13.

4 Symbols (and abbreviated terms)

4.1 Symbols specified in ISO/IEC 10118-1

This part of ISO/IEC 10118 makes use of the following symbols and notations defined in ISO/IEC 10118-1.

B_i A byte.

D Data.

H Hash-code.

IV Initializing value.

L_1 The length (in bits) of the first of the two input strings to the round-function Φ.

L_2 The 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.

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.

$X \oplus Y$ Exclusive-or of strings of bits X and Y (where $L_X = L_Y$).

4.2 Symbols specific to this part

For the purpose of this part of ISO/IEC 10118, the following symbols and notations apply:

a_i, a'_i Sequences of indices used in specifying a round-function.

A^i A sequence of constant matrices used in specifying the round-function defined in clause 13.

c_0 Function taking a string of 64 elements of GF(2^8) as input, and giving an 8 by 8 matrix with entries from GF(2^8) as output, used in specifying the round-function defined in clause 13.

c_1, c_2, c_3 Functions taking an 8 by 8 matrix of elements of GF(2^8) as input, and giving an 8 by 8 matrix with entries from GF(2^8) as output, used in specifying the round-function defined in clause 13.

c_4 Function taking two 8 by 8 matrices of elements of GF(2^8) as input, and giving an 8 by 8 matrix with entries from GF(2^8) as output, used in specifying the round-function defined in clause 13.

C_i, C'_i Constant words used in the round-functions.

C'' An 8 by 8 circulant matrix with entries chosen from GF(2^8) used in specifying the round-function in clause 13.

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.

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 by 8 matrix whose entries are chosen from GF(2^8).

q The number of blocks in the data string after the padding and splitting processes.

$R^n()$ The 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 places.

s A nonlinear substitution box, which replaces an element $x \in GF(2^8)$ with another element $s[x] \in GF(2^8)$;

$S^n()$ The 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.

$S'^n()$ The 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.

t_i, t'_i Shift-values used in specifying a round-function.

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.

Λ The bit-wise logical AND operation on bit-strings, i.e., if A, B are words then $A \Lambda B$ is the word equal to the bit-wise logical AND of A and B.

V The bit-wise logical OR operation on bit-strings, i.e., if A, B are words then $A \mathrm{V} B$ is the word equal to the bit-wise logical OR of A and B.

¬ The bit-wise logical NOT operation on a bit-string, i.e., if A is a word then $\neg A$ is the word equal to the bit-wise logical NOT of A.

$\uplus$ The modulo 2^w addition operation, where w is the number of bits in a word. I.e., if A and B are 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 1-4, defined in clauses 7-10, and 64 for dedicated hash-functions 5 and 6, defined in clauses 11 and 12.

- The multiplication operation of 8 by 8 matrices with entries chosen from $GF(2^8)$. I.e., if A and B are such matrices, then $A \bullet B$ 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 a multiplication of two of the entries as the remainder when a 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 a 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 word (or the matrix in clause 13) on the left side of the symbol shall be made 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 part of ISO/IEC 10118 shall select:

— one of the dedicated hash-functions specified below; and
— the length L_H of the hash-code H.

NOTE — The first and second dedicated hash-functions are defined so as to facilitate software implementations for 'little-endian' computers, i.e., where the lowest-addressed byte in a word is interpreted as the least significant; conversely, the third, fourth, fifth and sixth dedicated hash-functions are defined so as to facilitate software implementations for 'big-endian' computers, i.e., where the lowest-addressed byte in a word is interpreted as the most significant. However, by adjusting the definition appropriately, any of these six round-functions can be implemented on a 'big-endian' or a 'little-endian' computer. The seventh dedicated hash-function is defined to be 'endian-neutral', in the sense that it uses no endian-sensitive arithmetical operation (such as integer addition). If sequences of $GF(2^8)$ elements (i.e., bytes) are mapped to computer words to parallelize

such operations as exclusive-or, the byte disposition within a word is irrelevant, as long as the inverse mapping is consistent. All the hash-functions defined in this part of ISO/IEC 10118 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 — The choice of L_H affects the security of the hash-function. All of the hash-functions specified in this part of ISO/IEC 10118 are believed to be collision-resistant hash-functions in environments where performing $2^{L_H/2}$ hash-code computations is deemed to be computationally infeasible.

6 Model for dedicated hash-functions

The hash-functions specified in this part of ISO/IEC 10118 are based on the general model for hash-functions given in part 1 of this ISO/IEC standard, i.e., ISO/IEC 10118-1:2000.

In the specifications of the hash-functions in this part of ISO/IEC 10118, 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

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

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

The output transformation for the hash-functions specified in this part of ISO/IEC 10118 is that the hash-code H is derived by taking the leftmost L_H bits of the final L_2-bit output string H_q.

Identifiers are defined for each of the seven dedicated hash-functions specified in this standard. The hash-function identifiers for the dedicated hash-functions specified in clauses 7, 8, 9, 10, 11, 12 and 13 are equal to 31, 32, 33, 34, 35, 36 and 37 (hexadecimal) respectively. The range of values from 38 to 3F (hexadecimal) are reserved for future use as hash-function identifiers by this part of ISO/IEC 10118. The hash-function identifiers are also used in the OSI object identifiers assigned in Annex C.

7 Dedicated Hash-Function 1 (RIPEMD-160)

In this clause we specify a padding method, an initializing value, and a round-function for use in the general model for hash-functions described in ISO/IEC 10118-1:2000. 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 — Dedicated Hash-Function 1 defined in this clause is commonly called RIPEMD-160, [3].

7.1 Parameters, functions and constants

7.1.1 Parameters

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

7.1.2 Byte ordering convention

In the specification of the round-function of this clause 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

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

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 subclause 9.1.2.

7.1.3 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 ≤ i ≤ 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:

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

7.1.4 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:

$$\begin{aligned}
C_i &= \texttt{00000000}, && (0 \le i \le 15),\\
C_i &= \texttt{5A827999}, && (16 \le i \le 31),\\
C_i &= \texttt{6ED9EBA1}, && (32 \le i \le 47),\\
C_i &= \texttt{8F1BBCDC}, && (48 \le i \le 63),\\
C_i &= \texttt{A953FD4E}, && (64 \le i \le 79),
\end{aligned}$$

$$\begin{aligned}
C'_i &= \texttt{50A28BE6}, && (0 \le i \le 15),\\
C'_i &= \texttt{5C4DD124}, && (16 \le i \le 31),\\
C'_i &= \texttt{6D703EF3}, && (32 \le i \le 47),\\
C'_i &= \texttt{7A6D76E9}, && (48 \le i \le 63),\\
C'_i &= \texttt{00000000}, && (64 \le i \le 79).
\end{aligned}$$

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

Table 1

i	0	1	2	3	4	5	6	7
t_i	11	14	15	12	5	8	7	9
t'_i	8	9	9	11	13	15	15	5
a_i	0	1	2	3	4	5	6	7
a'_i	5	14	7	0	9	2	11	4

i	8	9	10	11	12	13	14	15
t_i	11	13	14	15	6	7	9	8
t'_i	7	7	8	11	14	14	12	6
a_i	8	9	10	11	12	13	14	15
a'_i	13	6	15	8	1	10	3	12

i	16	17	18	19	20	21	22	23
t_i	7	6	8	13	11	9	7	15
t'_i	9	13	15	7	12	8	9	11
a_i	7	4	13	1	10	6	15	3
a'_i	6	11	3	7	0	13	5	10

i	24	25	26	27	28	29	30	31
t_i	7	12	15	9	11	7	13	12
t'_i	7	7	12	7	6	15	13	11
a_i	12	0	9	5	2	14	11	8
a'_i	14	15	8	12	4	9	1	2

i	32	33	34	35	36	37	38	39
t_i	11	13	6	7	14	9	13	15
t'_i	9	7	15	11	8	6	6	14
a_i	3	10	14	4	9	15	8	1
a'_i	15	5	1	3	7	14	6	9

i	40	41	42	43	44	45	46	47
t_i	14	8	13	6	5	12	7	5
t'_i	12	13	5	14	13	13	7	5
a_i	2	7	0	6	13	11	5	12
a'_i	11	8	12	2	10	0	4	13

i	48	49	50	51	52	53	54	55
t_i	11	12	14	15	14	15	9	8
t'_i	15	5	8	11	14	14	6	14
a_i	1	9	11	10	0	8	12	4
a'_i	8	6	4	1	3	11	15	0

i	56	57	58	59	60	61	62	63
t_i	9	14	5	6	8	6	5	12
t'_i	6	9	12	9	12	5	15	8
a_i	13	3	7	15	14	5	6	2
a'_i	5	12	2	13	9	7	10	14

i	64	65	66	67	68	69	70	71
t_i	9	15	5	11	6	8	13	12
t'_i	8	5	12	9	12	5	14	6
a_i	4	0	5	9	7	12	2	10
a'_i	12	15	10	4	1	5	8	7

i	72	73	74	75	76	77	78	79
t_i	5	12	13	14	11	8	5	6
t'_i	8	13	6	5	15	13	11	11
a_i	14	1	3	8	11	6	15	13
a'_i	6	2	13	14	0	3	9	11

7.1.5 Initializing value

For this round-function the initializing value, *IV*, shall always be the following 160-bit string, represented here as a sequence of five words Y_0, Y_1, Y_2, Y_3, Y_4 in a hexadecimal representation, where Y_0 represents the left-most 32 of the 160 bits:

Y_0 = 67452301,
Y_1 = EFCDAB89,

Y_2 = `98BADCFE`,
Y_3 = `10325476`,
Y_4 = `C3D2E1F0`.

7.2 Padding method

The data string D needs to be padded to make it contain a number of bits which is an integer multiple of 512. The padding procedure operates as follows:

1. D is concatenated with a single '1' bit.

2. The result of the previous step is concatenated with between zero and 511 '0' bits such that the length (in bits) of the resultant string is congruent to 448 modulo 512. More explicitly, if the original length of D is L_D, and letting r be the remainder when L_D is divided by 512, then the number of concatenated zeros is equal to either 447-r (if $r \le 447$) or 959-r (if $r > 447$). The result will be a bit string whose length will be 64 bits short of an integer multiple of 512 bits.

3. Divide the 64-bit binary representation of L_D into two 32-bit strings, one representing the 'most significant half' of L_D and the other the 'least significant half'. Now concatenate the string resulting from the previous step with these two 32-bit strings, with the 'least significant half' preceding the 'most significant half'.

In the description of the round-function which follows, each 512-bit data block D_i, $1 \le i \le q$, is treated as a sequence of 16 words, Z_0, Z_1, …, Z_{15}, where Z_0 corresponds to the left-most 32 bits of D_i.

NOTE — The concatenation of the two 32-bit strings of L_D in step 3 is such that these two 32-bit strings are used directly as the words Z_{14} and Z_{15} of the last data block; based on the byte ordering convention in Clause 7.1.2, the least significant octet of L_D is the leftmost octet, and the most significant octet of L_D is the rightmost octet.

7.3 Description of the round-function

The round-function Φ operates as follows. Note that, in this description, we use the symbols W, X_0, X_1, X_2, X_3, X_4, X'_0, X'_1, X'_2, X'_3, X'_4 to denote eleven distinct words which contain values required in the computations.

1. Suppose the 512-bit (first) input to Φ is contained in Z_0, Z_1, …, Z_{15}, where Z_0 contains the left-most 32 of the 512 bits. Suppose also that the 160-bit (second) input to Φ is contained in five words, Y_0, Y_1, Y_2, Y_3, Y_4.

2. Let $X_0 := Y_0$, $X_1 := Y_1$, $X_2 := Y_2$, $X_3 := Y_3$ and $X_4 := Y_4$.

3. Let $X'_0 := Y_0$, $X'_1 := Y_1$, $X'_2 := Y_2$, $X'_3 := Y_3$ and $X'_4 := Y_4$.

4. For i := 0 to 79 do the following four steps in the order specified:

   (a) $W := S^{t_i}(X_0 \uplus g_i(X_1, X_2, X_3) \uplus Z_{a_i} \uplus C_i) \uplus X_4$;

   (b) $X_0 := X_4$; $X_4 := X_3$; $X_3 := S^{10}(X_2)$; $X_2 := X_1$; $X_1 := W$;

   (c) $W := S^{t'_i}(X'_0 \uplus g_{79-i}(X'_1, X'_2, X'_3) \uplus Z_{a'_i} \uplus C'_i) \uplus X'_4$;

   (d) $X'_0 := X'_4$; $X'_4 := X'_3$; $X'_3 := S^{10}(X'_2)$; $X'_2 := X'_1$; $X'_1 := W$.

5. Let

$$W := Y_0,$$
$$Y_0 := Y_1 \uplus X_2 \uplus X'_3,$$
$$Y_1 := Y_2 \uplus X_3 \uplus X'_4,$$
$$Y_2 := Y_3 \uplus X_4 \uplus X'_0,$$
$$Y_3 := Y_4 \uplus X_0 \uplus X'_1,$$
$$Y_4 := W \uplus X_1 \uplus X'_2.$$

6. The five words Y_0, Y_1, Y_2, Y_3, Y_4 represent the output of the round-function Φ. After the final iteration of the round-function, the five words Y_0, Y_1, Y_2, Y_3, Y_4 shall be converted to a sequence of 20 bytes using the inverse of the procedure specified in 7.1.2, and where Y_0 shall yield the first four bytes, Y_1 the next four bytes, and so on. Thus the first (left-most) byte will correspond to the least significant byte of Y_0, and the 20th (right-most) byte will correspond to the most significant byte of Y_4. The 20 bytes shall be converted to a string of 160 bits using the inverse of the procedure specified in clause 6, i.e., the first (left-most) bit will correspond to the most significant bit of the first (left-most) byte, and the 160th (right-most) bit will correspond to the least significant bit of the 20th (right-most) byte.

Figure 1 below shows steps a and b of item 4 of the round function Φ in Dedicated Hash-Function 1 (RIPEMD-160) (the other half, i.e., steps c and d is similar). In the round function Φ, steps a to c of item 4 are used for 80 times (i = 0, …, 79).

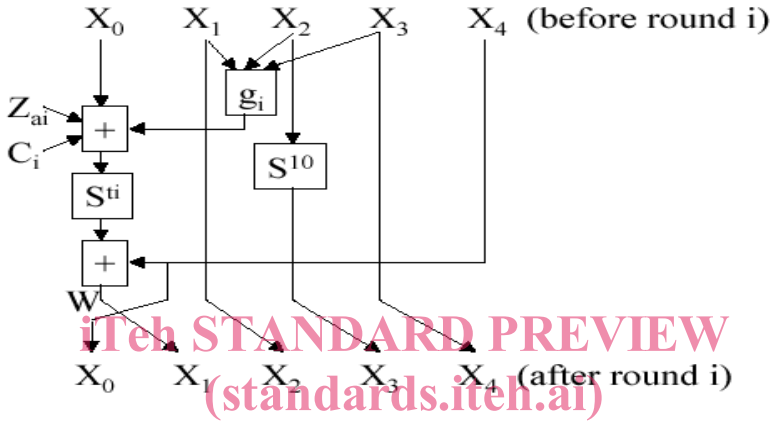

Figure 1. Part of the round function in Dedicated Hash-Function 1

8 Dedicated Hash-Function 2 (RIPEMD-128)

In this clause we specify a padding method, an initializing value, and a round-function for use in the general model for hash-functions described in ISO/IEC 10118-1:2000. The padding method, initializing value and round-function specified here, when used in the above general model, together define Dedicated Hash-Function 2. 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 2 is equal to 32 (hexadecimal).

NOTE — Dedicated Hash-Function 2 defined in this clause is commonly called RIPEMD-128, [3]. This hash-function should only be used in applications where a hash-code containing 128 bits or less is considered adequately secure.

8.1 Parameters, functions and constants

8.1.1 Parameters

For this hash-function L_1=512, L_2=128 and L_H is up to 128.

8.1.2 Byte ordering convention

The byte ordering convention for this hash-function is the same as that for the hash-function of clause 7.

8.1.3 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_{63} is used in this round-function, where each function g_i, $0 \le i \le 63$, 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 to be the same as the first 64 of the functions defined in subclause 7.1.3.

8.1.4 Constants

Two sequences of constant words C_0, C_1, …, C_{63} and C'_0, C'_1, …, C'_{63} 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 = 50A28BE6, ($0 \le i \le 15$),
C'_i = 5C4DD124, ($16 \le i \le 31$),
C'_i = 6D703EF3, ($32 \le i \le 47$),
C'_i = 00000000, ($48 \le i \le 63$).

Two sequences of 64 shift-values are also used in this round-function, where each shift-value is between 5 and 15. We denote these sequences by $(t_0, t_1, \ldots, t_{63})$ and $(t'_0, t'_1, \ldots, t'_{63})$, and they are defined to be equal to the first 64 values of the corresponding sequences defined in subclause 7.1.4.

Finally, two further sequences of 64 indices are used in this round-function, where each value in the sequence is between 0 and 15. We denote these sequences by $(a_0, a_1, \ldots, a_{63})$ and $(a'_0, a'_1, \ldots, a'_{63})$, and they are defined to be equal to the first 64 values of the corresponding sequences defined in subclause 7.1.4.

8.1.5 Initializing value

For this hash-function the initializing value, *IV*, shall always be the following 128-bit string, represented here as a sequence of four words Y_0, Y_1, Y_2, Y_3 in a hexadecimal representation, where Y_0 represents the left-most 32 of the 128 bits:

Y_0 = 67452301,
Y_1 = EFCDAB89,
Y_2 = 98BADCFE,
Y_3 = 10325476.

8.2 Padding method

The padding method to be used with this hash-function shall be the same as the padding method defined in subclause 7.2.

8.3 Description of the round-function

The round-function Φ operates as follows. Note that, in this description, we use the symbols W, X_0, X_1, X_2, X_3, X'_0, X'_1, X'_2, X'_3 to denote nine distinct words which contain values required in the computations.

1. Suppose the 512-bit (first) input to Φ is contained in Z_0, Z_1, …, Z_{15}, where Z_0 contains the left-most 32 of the 512 bits. Suppose also that the 128-bit (second) input to Φ is contained in four words, Y_0, Y_1, Y_2, Y_3.

2. Let $X_0 := Y_0$, $X_1 := Y_1$, $X_2 := Y_2$ and $X_3 := Y_3$.