INTERNATIONAL STANDARD

ISO/IEC 10118-3

First edition
1998-06-01

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 hachage dédiées

Reference number
ISO/IEC 10118-3:1998(E)

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 JTC1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

International Standard ISO/IEC 10118-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Sub-Committee SC27, *IT Security techniques.*

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

— *Part 1: General*

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

— *Part 3: Dedicated hash-functions*

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

Further parts may follow.

Annexes A, B, and C of this part of ISO/IEC 10118 are for information only.

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. Three distinct round-functions are specified, giving rise to distinct dedicated hash-functions. The first and third provide hash-codes of lengths up to 160 bits, and the second provides hash-codes of lengths up to 128 bits.

2 Normative reference

The following standard contains provisions which, through reference in the text, constitute provisions of this part of ISO/IEC 10118. At the time of publication, the edition indicated was valid. All standards are subject to revision and parties to agreements based on this part of ISO/IEC 10118 are encouraged to investigate the possibility of applying the most recent edition of the standard indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 10118–1: 1994, *Information technology — Security techniques — Hash-functions — Part 1: General.*

3 Definitions

For the purposes of this part of ISO/IEC 10118, the definitions given in ISO/IEC 10118–1 and the following definitions 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 hash-function identifier: A byte identifying a specific hash-function.

3.3 round-function: A function $\phi(.,.)$ that transforms two binary strings of lengths L_1 and L_2 to a binary string of length L_2. It is used iteratively as part of a hash-function, where it combines a data string of length L_1 with the previous output of length L_2.

3.4 word: A string of 32 bits.

4 Symbols and notation

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

D A data string to be input to the hash-function.

H Hash-code.

IV Initializing value.

L_X Length (in bits) of a bit-string X.

$X \oplus Y$ Exclusive-or of bit-strings X and Y.

For the purpose of this Part of ISO/IEC 10118, the following symbols and notation apply:

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

B_i A byte.

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

D_i A block derived from the data-string after the padding process.

f_i, g_i Functions taking 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.

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 IV.

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

$S^n()$ 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.

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.

ϕ A round-function, i.e. if $\mathbf{X}, \mathbf{Y}$ are bit-strings of lengths L_1 and L_2 respectively, then $\phi(\mathbf{X}, \mathbf{Y})$ is the string obtained by applying ϕ to $\mathbf{X}$ and $\mathbf{Y}$.

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

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

$\neg$ 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^{32} addition operation, i.e. if A, 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^{32}, where the result is constrained to lie between 0 and $2^{32} - 1$ inclusive.

$:=$ A symbol denoting the 'set equal to' operation used in procedural specifications of round-functions, where it indicates that the word 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 1 — 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 round-function is 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 the round-functions can be implemented on a 'big-endian' or a 'little-endian' computer. 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 2 — 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

6.1 General

The hash-functions specified in this standard require the use of a round-function ϕ. In subsequent clauses of this part of ISO/IEC 10118, three alternatives for the function ϕ are specified.

The hash-functions which are specified in this standard provide hash-codes of length L_H, where L_H is less than or equal to the value of L_2 for the round-function ϕ being used.

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, $x_0, x_1, \ldots, x_{8n-1}$, then it shall be interpreted as a sequence of n bytes, $B_0, B_1, \ldots, B_{n-1}$, 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 $(0 \leq i < n)$.

Identifiers are defined for each of the three dedicated hash-functions specified in this standard. The hash-function identifiers for the dedicated hash-functions specified in clauses 7, 8 and 9 are equal to 31, 32, and 33 (hexadecimal) respectively. The range of values from 34 to 3F (hexadecimal) are reserved for future use as hash-function identifiers by this part of ISO/IEC 10118.

6.2 Hashing operation

Let ϕ be a round-function and IV be an initializing value of length L_2. For the hash-functions specified in this part of ISO/IEC 10118, the value of the IV shall be fixed for a given round-function ϕ.

The hash-code H of the data D shall be calculated in four steps.

6.2.1 Step 1 (padding)

The data string D is padded in order to ensure that its length is a multiple of L_1. Specific instances of padding methods are specified in subsequent clauses of this part of ISO/IEC 10118.

6.2.2 Step 2 (splitting)

The padded version of the data string D is split into L_1-bit blocks $D_1, D_2, \ldots, D_q$, where D_1 represents the first L_1 bits of the padded version of D, D_2 represents the next L_1 bits, and so on. The Padding and Splitting Processes are illustrated in Figure 1.

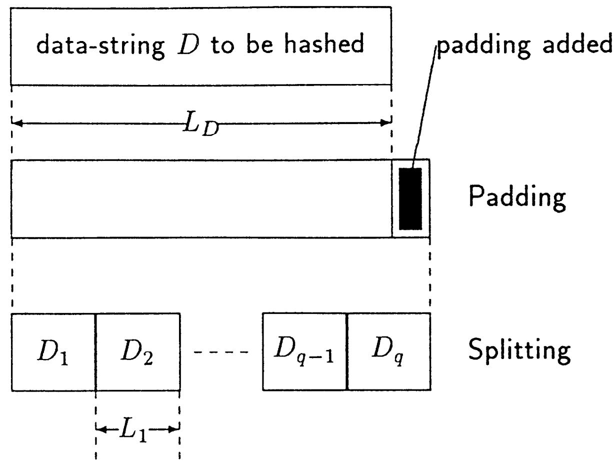

Figure 1: **Padding & splitting processes**

6.2.3 Step 3 (iteration)

Let $D_1, D_2, \ldots, D_q$ be the L_1-bit blocks of the data after padding and splitting. Let H_0 be a bit-string equal to IV. The L_2-bit strings $H_1, H_2, \ldots, H_q$ are calculated iteratively in the following way.

for i from 1 to q:

$$H_i = \phi(D_i, H_{i-1});$$

The Iteration Process is illustrated in Figure 2.

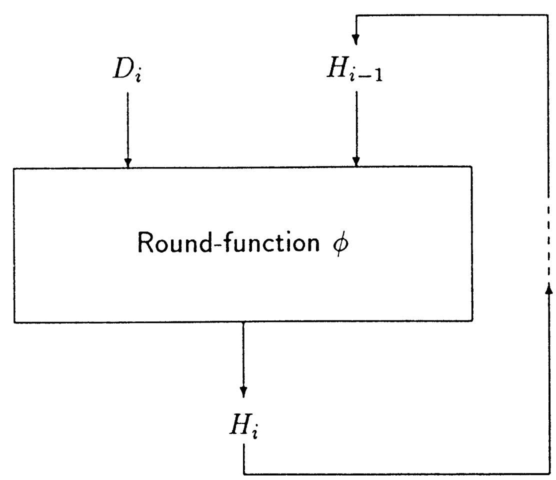

Figure 2: **The Iteration Process**

6.2.4 Step 4 (truncation)

The hash-code H is derived by taking the leftmost L_H bits of the final L_2-bit output string H_q.

7 Dedicated Hash-Function 1

NOTE — This clause contains a description of the round-function, initializing value and padding method for RIPEMD–160, [3].

7.1 General

In this clause we specify a padding method, an initializing value, and a round-function for use in the general model described in this part of ISO/IEC 10118. 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).

7.2 Parameters, functions and constants

7.2.1 Parameters

For this hash-function $L_1 = 512$ and $L_2 = 160$.

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 words, each 512-bit block being made up of 16 such words. A sequence of 64 bytes, $B_0, B_1, \ldots, B_{63}$, shall be interpreted as a sequence of 16 words, $Z_0, Z_1, \ldots, Z_{15}$, 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 subclause 9.2.2.

7.2.3 Functions

To facilitate software implementation, the round-function ϕ is described in terms of operations on words. A sequence of functions $g_0, g_1, \ldots, g_{79}$ 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:

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

7.2.4 Constants

Two sequences of constant words $C_0, C_1, \ldots, C_{79}$ and $C'_0, C'_1, \ldots, C'_{79}$ 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 the following table.

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

7.2.5 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:

$$
\begin{aligned}
Y_0 &= \texttt{67452301}, \\
Y_1 &= \texttt{EFCDAB89}, \\
Y_2 &= \texttt{98BADCFE}, \\
Y_3 &= \texttt{10325476}, \\
Y_4 &= \texttt{C3D2E1F0}.
\end{aligned}
$$

7.3 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 $r \leq 447$) 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, $1 \leq i \leq q$, is treated as a sequence of 16 words, $Z_0, Z_1, \ldots, Z_{15}$, where Z_0 corresponds to the left-most 32 bits of D_i.

7.4 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 $Z_0, Z_1, \ldots, Z_{15}$, 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

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

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 7.2.2, 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 corespond 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 6.1, 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.

8 Dedicated Hash-Function 2

NOTE — This clause contains a description of the round-function, initializing value and padding method for 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 General

In this clause we specify a padding method, an initializing value, and a round-function for use in the general model described in this part of ISO/IEC 10118. 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).

8.2 Parameters, functions and constants

8.2.1 Parameters

For this hash-function $L_1 = 512$ and $L_2 = 128$.

8.2.2 Byte ordering convention

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

8.2.3 Functions

To facilitate software implementation, the round-function ϕ is described in terms of operations on words. A sequence of functions $g_0, g_1, \ldots, g_{63}$ is used in this round-function, where each function g_i, $0 \leq i \leq 63$, 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 7.2.3.

8.2.4 Constants

Two sequences of constant words $C_0, C_1, \ldots, C_{63}$ and $C'_0, C'_1, \ldots, C'_{63}$ 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:

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

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

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 7.2.4.

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 7.2.4.

8.2.5 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:

$$
\begin{aligned}
Y_0 &= \texttt{67452301}, \\
Y_1 &= \texttt{EFCDAB89}, \\
Y_2 &= \texttt{98BADCFE}, \\
Y_3 &= \texttt{10325476}.
\end{aligned}
$$

8.3 Padding method

The padding method to be used with this hash-function shall be the same as the padding method defined in subclause 7.3.

8.4 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 $Z_0, Z_1, \ldots, Z_{15}$, 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 $X_3 := Y_3$.
3. Let $X'_0 := Y_0$, $X'_1 := Y_1$, $X'_2 := Y_2$ and $X'_3 := Y_3$.
4. For $i := 0$ to 63 do the following four steps in the order specified:
   (a) $W := S^{t_i}(X_0 \boxplus g_i(X_1, X_2, X_3) \boxplus Z_{a_i} \boxplus C_i)$;
   (b) $X_0 := X_3$; $X_3 := X_2$; $X_2 := X_1$; $X_1 := W$;
   (c) $W := S^{t'_i}(X'_0 \boxplus g_{63-i}(X'_1, X'_2, X'_3) \boxplus Z_{a'_i} \boxplus C'_i)$;
   (d) $X'_0 := X'_3$; $X'_3 := X'_2$; $X'_2 := X'_1$; $X'_1 := W$;
5. Let

$$
\begin{aligned}
W &:= Y_0, \\
Y_0 &:= Y_1 \boxplus X_2 \boxplus X'_3, \\
Y_1 &:= Y_2 \boxplus X_3 \boxplus X'_0, \\
Y_2 &:= Y_3 \boxplus X_0 \boxplus X'_1, \\
Y_3 &:= W \boxplus X_1 \boxplus X'_2.
\end{aligned}
$$

6. The four words Y_0, Y_1, Y_2, Y_3 represent the output of the round-function ϕ. After the final iteration of the round-function, the four words Y_0, Y_1, Y_2, Y_3 shall be converted to a sequence of 16 bytes using the inverse of the procedure specified in 7.2.2, 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 16th (right-most) byte will corespond to the most significant byte of Y_3. The 16 bytes shall be converted to a string of 128 bits using the inverse of the procedure specified in 6.1, i.e. the

first (left-most) bit will correspond to the most significant bit of the first (left-most) byte, and the 128th (right-most) bit will correspond to the least significant bit of the 16th (right-most) byte.

9 Dedicated Hash-Function 3

NOTE — This clause contains a description of the round-function, initializing value and padding method for SHA–1 (the US NIST 'Secure Hash Algorithm'), [2].

9.1 General

In this clause we specify a padding method, an initializing value, and a round-function for use in the general model described in this part of ISO/IEC 10118. The padding method, initializing value and round-function specified here, when used in the above general model, together define Dedicated Hash-Function 3. 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 3 is equal to 33 (hexadecimal).

9.2 Parameters, functions and constants

9.2.1 Parameters

For this hash-function $L_1 = 512$ and $L_2 = 160$.

9.2.2 Byte ordering convention

In the specification of the round-function of clause 9 it is assumed that the block input to the round-function is in the form of a sequence of words, each 512-bit block being made up of 16 such words. A sequence of 64 bytes, $B_0, B_1, \ldots, B_{63}$, shall be interpreted as a sequence of 16 words, $Z_0, Z_1, \ldots, Z_{15}$, in the following way. Each group of four consecutive bytes is considered as a word, the first byte of a word being the most significant byte of that word. Hence

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

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

NOTE — The byte-ordering specified here is different from that of subclause 7.2.2.

9.2.3 Functions

To facilitate software implementation, the round-function ϕ is described in terms of operations on words. A sequence of functions $f_0, f_1, \ldots, f_{79}$ is used in this round-function, where each function f_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 f_i are defined as follows:

$$\begin{aligned}
f_i(X_0,X_1,X_2) &= (X_0 \wedge X_1) \vee (\neg X_0 \wedge X_2), \\
&\quad (0 \le i \le 19), \\
f_i(X_0,X_1,X_2) &= X_0 \oplus X_1 \oplus X_2, \quad (20 \le i \le 39), \\
f_i(X_0,X_1,X_2) &= (X_0 \wedge X_1) \vee (X_0 \wedge X_2) \vee (X_1 \wedge X_2), \\
&\quad (40 \le i \le 59), \\
f_i(X_0,X_1,X_2) &= X_0 \oplus X_1 \oplus X_2, \quad (60 \le i \le 79).
\end{aligned}$$

9.2.4 Constants

A sequence of constant words $C_0, C_1, \ldots, C_{79}$ is 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:

$$\begin{aligned}
C_i &= \texttt{5A827999}, \quad (0 \le i \le 19), \\
C_i &= \texttt{6ED9EBA1}, \quad (20 \le i \le 39), \\
C_i &= \texttt{8F1BBCDC}, \quad (40 \le i \le 59), \\
C_i &= \texttt{CA62C1D6}, \quad (60 \le i \le 79).
\end{aligned}$$

9.2.5 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:

$$\begin{aligned}
Y_0 &= \texttt{67452301}, \\
Y_1 &= \texttt{EFCDAB89}, \\
Y_2 &= \texttt{98BADCFE}, \\
Y_3 &= \texttt{10325476}, \\
Y_4 &= \texttt{C3D2E1F0}.
\end{aligned}$$

9.3 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 $r \leq 447$) 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. Concatenate the string resulting from the previous step with the 64-bit binary representation of L_D, most significant bit first.

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

9.4 Description of the round-function

The round-function ϕ operates as follows. Note that, in this description, we use the symbols $W, X_0, X_1, X_2, X_3, X_4, Z_0, Z_1, \ldots, Z_{79}$ to denote 86 distinct words which contain values required in the computations.

1. Suppose the 512-bit (first) input to ϕ is contained in $Z_0, Z_1, \ldots, Z_{15}$, 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. For $i = 16$ to 79 let

$$Z_i := S^1(Z_{i-3} \oplus Z_{i-8} \oplus Z_{i-14} \oplus Z_{i-16}).$$

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 two steps

   (a) $W := S^5(X_0) \uplus f_i(X_1, X_2, X_3) \uplus X_4 \uplus Z_i \uplus C_i$;

   (b) $X_4 := X_3$; $X_3 := X_2$; $X_2 := S^{30}(X_1)$; $X_1 := X_0$; $X_0 := W$.

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

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 9.2.2, 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 most significant byte of Y_0, and the 20th (right-most) byte will corespond to the least 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 6.1, 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.

Annex A
(informative)

Examples

A.1 General

This annex gives examples for the computation of Dedicated Hash-Functions 1, 2 and 3. Nine examples of hash-code calculation are given for each of the hash-functions. For each of the hash-functions, intermediate values derived during the hash-function's operation are given for examples numbers 3 and 8.

A.2 Dedicated Hash-Function 1

Throughout this annex we refer to ASCII coding of data strings; this is equivalent to coding using ISO 646.

NOTE — Reference [3] contains a pseudocode description of Dedicated Hash-Function 1.

A.2.1 Example 1

In this example the data-string is the empty string, i.e. the string of length zero.

The hash-code is the following 160-bit string.


```
9C 11 85 A5 C5 E9 FC 54 61 28 08 97 7E E8 F5 48 B2 25 8D 31
```


A.2.2 Example 2

In this example the data-string consists of a single byte, namely the ASCII-coded version of the letter 'a'.

The hash-code is the following 160-bit string.

```
0B DC 9D 2D 25 6B 3E E9 DA AE 34 7B E6 F4 DC 83 5A 46 7F FE
```

A.2.3 Example 3

In this example the data-string is the three-byte string consisting of the ASCII-coded version of 'abc'. This is equivalent to the bit-string: '01100001 01100010 01100011'.

After the padding process, the single 16-word block derived from the data-string is as follows.

80636261	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000018	00000000

The following are (hexadecimal representations of) the successive values of the variables X_0, X_1, X_2, X_3, X_4, X'_0, X'_1, X'_2, X'_3, X'_4.

```
67452301, EFCDAB89, 98BADCFE, 10325476, C3D2E1F0, 67452301, EFCDAB89, 98BADCFE, 10325476, C3D2E1F0
C3D2E1F0, 3115FC67, EFCDAB89, EB73FA62, 10325476, C3D2E1F0, DDD63FB8, EFCDAB89, EB73FA62, 10325476
10325476, B41192D5, 3115FC67, 36AE27BF, EB73FA62, 10325476, 322E7AE3, DDD63FB8, 36AE27BF, EB73FA62
EB73FA62, 3A35DC50, B41192D5, 57F19CC4, 36AE27BF, EB73FA62, 883EE903, 322E7AE3, 58FEE377, 36AE27BF
```

36AE27BF, D3786413, 3A35DC50, 464B56D0, 57F19CC4, 36AE27BF, 92B2B79B, 883EE903, B9EB8CC8, 58FEE377
57F19CC4, 0E946720, D3786413, D77140E8, 464B56D0, 58FEE377, F9091FF2, 92B2B79B, FBA40E20, B9EB8CC8
464B56D0, D52BF632, 0E946720, E1904F4D, D77140E8, B9EB8CC8, E5B09992, F9091FF2, CADE6E4A, FBA40E20
D77140E8, 150BD8A8, D52BF632, 519C803A, E1904F4D, FBA40E20, 8B2D9FB3, E5B09992, 247FCBE4, CADE6E4A
E1904F4D, 3D6F601F, 150BD8A8, AFD8CB54, 519C803A, CADE6E4A, E755F422, 8B2D9FB3, C2664B96, 247FCBE4
519C803A, B7B60384, 3D6F601F, 2F62A054, AFD8CB54, 247FCBE4, 5922D09E, E755F422, B67ECE2C, C2664B96
AFD8CB54, B85A0A3F, B7B60384, BD807CF5, 2F62A054, C2664B96, CF24E72C, 5922D09E, 57D08B9D, B67ECE2C
2F62A054, 7F8B38E5, B85A0A3F, D80E12DE, BD807CF5, B67ECE2C, CA6A1C75, CF24E72C, 8B427964, 57D08B9D
BD807CF5, 9DACA495, 7F8B38E5, 6828FEE1, D80E12DE, 57D08B9D, 227F6D84, CA6A1C75, 939CB33C, 8B427964
D80E12DE, BC05F46F, 9DACA495, 2CE395FE, 6828FEE1, 8B427964, 5D801685, 227F6D84, A871D729, 939CB33C
6828FEE1, 1494F053, BC05F46F, B2925676, 2CE395FE, 939CB33C, B3C3F4D5, 5D801685, FDB61089, A871D729
2CE395FE, 85861D02, 1494F053, 17D1BEF0, B2925676, A871D729, 3D16242D, B3C3F4D5, 005A1576, FDB61089
B2925676, 597BF629, 85861D02, 53C14C52, 17D1BEF0, FDB61089, FF459078, 3D16242D, 0FD356CF, 005A1576
17D1BEF0, 6347EF78, 597BF629, 18740A16, 53C14C52, 005A1576, 927E40A8, FF459078, 5890B4F4, 0FD356CF
53C14C52, 45C8FA44, 6347EF78, EFD8A565, 18740A16, 0FD356CF, ACBB994E, 927E40A8, 1641E3FD, 5890B4F4
18740A16, AD2956AF, 45C8FA44, 1FBDE18D, EFD8A565, 5890B4F4, AD30AD24, ACBB994E, F902A249, 1641E3FD
EFD8A565, 5EAF16B7, AD2956AF, 23E91117, 1FBDE18D, 1641E3FD, 6261732E, AD30AD24, EE653AB2, F902A249
1FBDE18D, 41730D4B, 5EAF16B7, A55ABEB4, 23E91117, F902A249, 45ED27AF, 6261732E, C2B492B4, EE653AB2
23E91117, FC0CCBD3, 41730D4B, BC5ADD7A, A55ABEB4, EE653AB2, 243C5668, 45ED27AF, 85CCB989, C2B492B4
A55ABEB4, 042ECC93, FC0CCBD3, CC352D05, BC5ADD7A, C2B492B4, 82F89BD1, 243C5668, B49EBD17, 85CCB989
BC5ADD7A, 4D4D4377, 042ECC93, 332F4FF0, CC352D05, 85CCB989, 5FC74686, 82F89BD1, F159A090, B49EBD17
CC352D05, 5207002B, 4D4D4377, BB324C10, 332F4FF0, B49EBD17, B2720031, 5FC74686, E26F460B, F159A090
332F4FF0, 388278F5, 5207002B, 350DDD35, BB324C10, F159A090, 58A100F8, B2720031, 1D1A197F, E26F460B
BB324C10, 62879D70, 388278F5, 1C00AD48, 350DDD35, E26F460B, 5992068B, 58A100F8, C800C6C9, 1D1A197F
350DDD35, A30A1FD9, 62879D70, 09E3D4E2, 1C00AD48, 1D1A197F, CC290DCA, 5992068B, 8403E162, C800C6C9
1C00AD48, BDA2B31B, A30A1FD9, 1E75C18A, 09E3D4E2, C800C6C9, 863D625E, CC290DCA, 481A2D66, 8403E162
09E3D4E2, F7211DEE, BDA2B31B, 287F668C, 1E75C18A, 8403E162, 6061B5A5, 863D625E, A4372B30, 481A2D66
1E75C18A, B6A665C6, F7211DEE, 8ACC6EF6, 287F668C, 481A2D66, AA98ADB5, 6061B5A5, F5897A18, A4372B30
287F668C, 2D30FA02, B6A665C6, 8477BBDC, 8ACC6EF6, A4372B30, 2999255A, AA98ADB5, 86D69581, F5897A18
8ACC6EF6, C76D12F9, 2D30FA02, 99971ADA, 8477BBDC, F5897A18, 98237631, 2999255A, 62B6D6AA, 86D69581
8477BBDC, 516F84DF, C76D12F9, C3E808B4, 99971ADA, 86D69581, 6C472A90, 98237631, 649568A6, 62B6D6AA
99971ADA, F3FA5B05, 516F84DF, B44BE71D, C3E808B4, 62B6D6AA, 2EAD5672, 6C472A90, 8DD8C660, 649568A6
C3E808B4, D539625E, F3FA5B05, BE137D45, B44BE71D, 649568A6, C5CB48BA, 2EAD5672, 1CAA41B1, 8DD8C660
B44BE71D, D8500C99, D539625E, E96C17CF, BE137D45, 8DD8C660, 05286DFB, C5CB48BA, B559C8BA, 1CAA41B1
BE137D45, 7ECDE5B2, D8500C99, E5897B54, E96C17CF, 1CAA41B1, 88396DD2, 05286DFB, 2D22EB17, B559C8BA
E96C17CF, 681D30B9, 7ECDE5B2, 40326761, E5897B54, B559C8BA, 333F2212, 88396DD2, A1B7EC14, 2D22EB17
E5897B54, 960F7BFD, 681D30B9, 3796C9FB, 40326761, 2D22EB17, C699295B, 333F2212, E5B74A20, A1B7EC14
40326761, 6770E498, 960F7BFD, 74C2E5A0, 3796C9FB, A1B7EC14, BFD68874, C699295B, FC8848CC, E5B74A20
3796C9FB, 75EB06C5, 6770E498, 3DEFF658, 74C2E5A0, E5B74A20, BDDF3474, BFD68874, 64A56F1A, FC8848CC
74C2E5A0, 14FA827A, 75EB06C5, C392619D, 3DEFF658, FC8848CC, 8CBC87E9, BDDF3474, 5A21D2FF, 64A56F1A
3DEFF658, 804B0068, 14FA827A, AC1B15D7, C392619D, 64A56F1A, CDDA6EBF, 8CBC87E9, 7CD1D2F7, 5A21D2FF
C392619D, 475BA81B, 804B0068, EA09E853, AC1B15D7, 5A21D2FF, 656C7DA3, CDDA6EBF, F21FA632, 7CD1D2F7
AC1B15D7, D26BC25D, 475BA81B, 2C01A201, EA09E853, 7CD1D2F7, 76D66CA3, 656C7DA3, 69BAFF37, F21FA632
EA09E853, DBC5A2CB, D26BC25D, 6EA06D1D, 2C01A201, F21FA632, C9B17F72, 76D66CA3, B1F68D95, 69BAFF37
2C01A201, 77367F5E, DBC5A2CB, AF097749, 6EA06D1D, 69BAFF37, 65A60151, C9B17F72, 59B28DDB, B1F68D95
6EA06D1D, 8155A6B4, 77367F5E, 168B2F6F, AF097749, B1F68D95, 33F3AC81, 65A60151, C5FDCB26, 59B28DDB
AF097749, C90C4D38, 8155A6B4, D9FD79DC, 168B2F6F, 59B28DDB, 9BFB827D, 33F3AC81, 98054596, C5FDCB26
168B2F6F, 9762713B, C90C4D38, 569AD205, D9FD79DC, C5FDCB26, DDC8130E, 9BFB827D, CEB204CF, 98054596
D9FD79DC, 7EBF9C32, 9762713B, 3134E324, 569AD205, 98054596, C24C2C79, DDC8130E, EE09F66F, CEB204CF
569AD205, 20EFFA01, 7EBF9C32, 89C4EE5D, 3134E324, CEB204CF, F255847E, C24C2C79, 204C3B77, EE09F66F
3134E324, 75B7117F, 20EFFA01, FE70C9FA, 89C4EE5D, EE09F66F, DCD63949, F255847E, 30B1E709, 204C3B77
89C4EE5D, A96BE4C7, 75B7117F, BFE80483, FE70C9FA, 204C3B77, 5B99238D, DCD63949, 5611FBC9, 30B1E709
FE70C9FA, 5E3201FC, A96BE4C7, DC45FDD6, BFE80483, 30B1E709, B43484F4, 5B99238D, 58E52773, 5611FBC9
BFE80483, 2CF95A98, 5E3201FC, AF931EA5, DC45FDD6, 5611FBC9, 52325A09, B43484F4, 648E356E, 58E52773
DC45FDD6, 1393F0C3, 2CF95A98, C807F178, AF931EA5, 58E52773, D015577D, 52325A09, D213D2D0, 648E356E
AF931EA5, BB49CCF7, 1393F0C3, E56A60B3, C807F178, 648E356E, BB9C87C4, D015577D, C9682548, D213D2D0
C807F178, 6A330EB4, BB49CCF7, 4FC30C4E, E56A60B3, D213D2D0, B1BB1A2E, BB9C87C4, 555DF740, C9682548
E56A60B3, 14E58204, 6A330EB4, 2733DEED, 4FC30C4E, C9682548, AC77F96D, B1BB1A2E, 721F12EE, 555DF740
4FC30C4E, 79AAF53E, 14E58204, CC3AD1A8, 2733DEED, 555DF740, 1774D326, AC77F96D, EC68BAC6, 721F12EE
2733DEED, 210769B3, 79AAF53E, 96081053, CC3AD1A8, 721F12EE, A625F112, 1774D326, DFE5B6B1, EC68BAC6
CC3AD1A8, F44B53A7, 210769B3, ABD4F9E6, 96081053, EC68BAC6, 5DCA4D12, A625F112, D34C985D, DFE5B6B1