INTERNATIONAL STANDARD

ISO/IEC 10118-2

Second edition
2000-12-15

Information technology — Security techniques — Hash-functions —

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

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

Partie 2: Fonctions de brouillage utilisant un chiffrement par blocs de n *bits*

Reference number
ISO/IEC 10118-2:2000(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.ch
Web www.iso.ch

Printed 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 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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. 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 10118-2 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 10118-2:1994), which has been technically revised to conform to the general model described in ISO/IEC 10118-1, and to add two additional hash-functions. Note, however, that implementations which comply with ISO/IEC 10118-2:1994 will be compliant with this edition of ISO/IEC 10118-2.

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*

— *Part 3: Dedicated hash-functions*

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

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

Introduction

The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this part of ISO/IEC 10118 may involve the use of a patent concerning the “Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function,” (U.S. Patent 4,908,861 issued 1990-03-13).

ISO and IEC take no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured ISO and IEC that he is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with ISO and IEC. Information may be obtained from:

Director of Licensing
International Business Machnies Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 10118 may be the subject of patent rights other than those identified above. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Information technology — Security techniques — Hash-functions —

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

1 Scope

This part of ISO/IEC 10118 specifies hash-functions which make use of an n-bit block cipher algorithm. They are therefore suitable for an environment in which such an algorithm is already implemented.

Four hash-functions are specified. The first provides hash-codes of length smaller than or equal to n, where n is the block-length of the algorithm used. The second provides hash-codes of length less than or equal to $2n$; the third provides hash-codes of length equal to $2n$; and the fourth provides hash-codes of length $3n$. All four of the hash-functions specified in this part of ISO/IEC 10118 conform to the general model specified in ISO/IEC 10118-1.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 10118. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 10118 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 10116:1997, *Information technology — Security techniques — Modes of operation for an n-bit block cipher.*

ISO/IEC 10118-1:2000, *Information technology — Security techniques — Hash-functions — Part 1: General.*

3 Terms and definitions

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

3.1
***n*-bit block cipher**
a block cipher with the property that plaintext blocks and ciphertext blocks are n bits in length (see ISO/IEC 10116)

4 Symbols and abbreviated terms

For the purposes of this part of ISO/IEC 10118, the symbols and abbreviations given in ISO/IEC 10118-1 and the following apply:

e — n-bit block cipher algorithm (see ISO/IEC 10116)

K — Key for the algorithm e (see ISO/IEC 10116)

$e_K(P)$ Operation of encipherment using the algorithm e and the key K (see ISO/IEC 10116) on plaintext P

u or u' Transformation of one n-bit block into a key for the algorithm e

B^L When n is even, the string composed of the $n/2$ leftmost bits of the block B. When n is odd, the string composed of the $(n+1)/2$ leftmost bits of the block B

B^R When n is even, the string composed of the $n/2$ rightmost bits of the block B. When n is odd, the string composed of the $(n-1)/2$ rightmost bits of the block B

B^x When B is a string of n m-bit blocks, B^x represents the x-th m-bit block of B

$B^{x\text{-}y}$ When B is a string of n m-bit blocks, $B^{x\text{-}y}$ represents the x-th through the y-th m-bit blocks of B

5 Use of the general model

The hash-functions specified in the next four clauses provide hash-codes H of length L_H. The hash-function conforms to the general model specified in ISO/IEC 10118-1. For each of the four hash-functions that follow, it is therefore only necessary to specify:

— the parameters L_1, L_2;

— the padding method;

— the initializing value IV;

— the round-function ϕ;

— the output transformation T.

The use of a hash-function defined using the general model will also require the selection of the parameter L_H.

6 Hash-function one

6.1 Parameter selection

The parameters L_1 and L_2 and L_H for the hash-function specified in this clause shall satisfy $L_1 = L_2 = n$, and L_H is less than or equal to n.

6.2 Padding method

The selection of the padding method for use with this hash-function is beyond the scope of this part of ISO/IEC 10118. Examples of padding methods are presented in annex A of ISO/IEC 10118-1:2000.

6.3 Initializing value

The selection of the IV for use with this hash-function is beyond the scope of this part of ISO/IEC 10118. The value of the IV shall be agreed upon and fixed by the users of the hash-function.

6.4 Round-function

The round-function ϕ combines a padded data block D_i (of $L_1 = n$-bits) with H_{i-1}, the previous output of the round-function (of $L_2 = n$ bits), to yield H_i. As part of the round-function it is necessary to choose a function u, which transforms an n-bit block into a key for use with the block cipher algorithm e. The selection of the function u for use with this hash-function is outside the scope of this part of ISO/IEC 10118 (see annex A for guidance).

The round-function itself is defined as follows:

$$\phi(D_j, H_{j-1}) = e_{K_j}(D_j) \oplus D_j$$

where $K_j = u(H_{j-1})$. The round-function is shown in Figure 1.

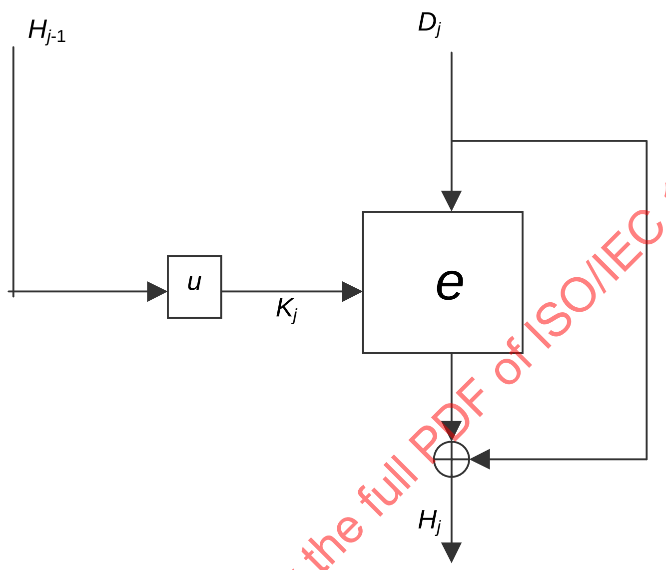

Figure 1 — Round-function of hash-function one

6.5 Output transformation

The output transformation T is simply truncation, i.e., the hash-code H is derived by taking the leftmost L_H bits of the final output block H_q.

7 Hash-function two

7.1 Parameter selection

The parameters L_1 and L_2 and L_H for the hash-function specified in this clause shall satisfy $L_1 = n$, $L_2 = 2n$, and L_H is less than or equal to $2n$.

7.2 Padding method

The selection of the padding method for use with this hash-function is beyond the scope of this part of ISO/IEC 10118. Examples of padding methods are presented in annex A of ISO/IEC 10118-1:2000.

7.3 Initializing value

The selection of the IV (of length $2n$) for use with this hash-function is beyond the scope of this part of ISO/IEC 10118. The value of the IV shall be agreed upon and fixed by the users of the hash-function. However, the IV shall be selected such that $u(IV^L)$ and $u'(IV^R)$ are different.

7.4 Round-function

The round-function ϕ combines a padded data block D_i (of $L_1 = n$ bits) with $H_{i\text{-}1}$, the previous output of the round-function (of $L_2 = 2n$ bits), to yield H_i. As part of the round-function it is necessary to choose two transformations u and u'. These transformations are used to transform an output block into two suitable L_K bit keys for the algorithm e. The specification of u and u' is beyond the scope of this part of ISO/IEC 10118. However, it should be taken into consideration that the selection of u and u' is important for the security of the hash-function (see annex A).

Set H_0^L and H_0^R equal to IV^L and IV^R respectively. The output blocks are calculated iteratively in the following way, for j = 1 to q:

$\phi\,(D_j, H_{j\text{-}1}) = H_j$

$K_j^L = u(H_{j\text{-}1}^{\;L})$ and $K_j^R = u'\,(H_{j\text{-}1}^{\;R})$

$B_j = e_{K_j^L}(D_j) \oplus D_j$, and $B'_j = e_{K_j^R}(D_j) \oplus D_j$

$H_j^L = B_j^L \,||\, B'^R_j$ and $H_j^R = B'^L_j \,||\, B_j^R$

The round-function is shown in Figure 2.

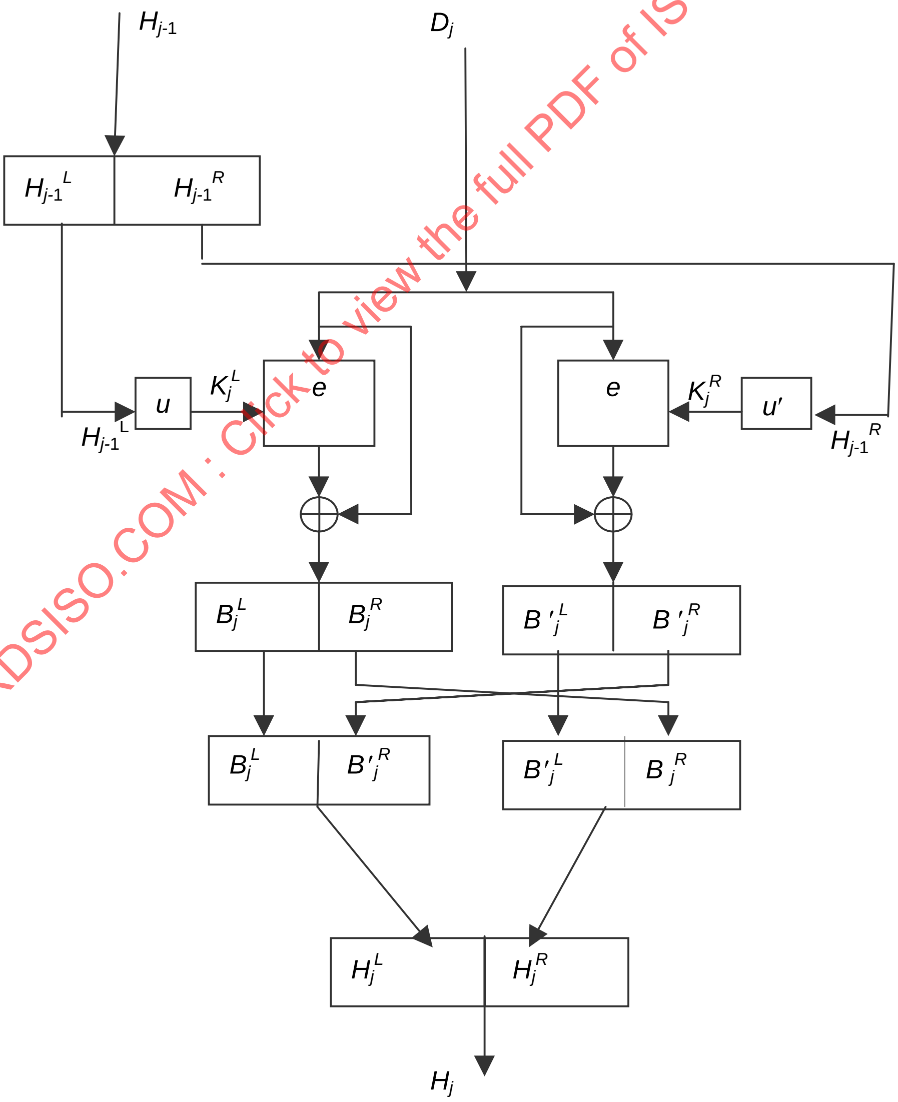

Figure 2 — Round-function of hash-function two

7.5 Output transformation

If L_H is even, the hash-code is the concatenation of the $L_H/2$ leftmost bits of H_q^L and the $L_H/2$ leftmost bits of H_q^R. If L_H is odd, the hash-code is the concatenation of the $(L_H+1)/2$ leftmost bits of H_q^L and the $(L_H-1)/2$ leftmost bits of H_q^R.

8 Hash-function three

The hash-function specified in this clause provides hash-codes of length L_H, where L_H is equal to $2n$ for even values of n.

8.1 General

Some specific definitions that are required to specify hash-function three follow.

Transformation u:

Define r mappings u_1, u_2, …, u_r from the ciphertext space to the key space, such that,

For all i, j from the set $\{1,2,\ldots,r\}$, $j \neq i$, $u_i(C) \neq u_j(C)$ for all values of C.

This can be achieved by fixing specific key bits: e.g., if $r = 8$ one can fix three key bits to the values 000, 001, ..., 111. Additional conditions might be imposed upon the mappings u_i, for example, to avoid the problems related to weak keys or complementation properties of the block cipher.

Function f_i:

Define the r functions f_i as follows:

$$f_i(X,Y) = e_{ui(X)}(Y) \oplus Y, 1 \leq i \leq r.$$

Linear mapping β:

Define the linear mapping β that takes as input a $2n$-bit string $X = x_0||x_1||x_2||x_3$ and maps it to a $2n$-bit string $Y = y_0||y_1||y_2||y_3$ as follows:

$$y_0 := x_0 \oplus x_3$$

$$y_1 := x_0 \oplus x_1 \oplus x_3$$

$$y_2 := x_1 \oplus x_2$$

$$y_3 := x_2 \oplus x_3$$

Here x_i and y_j are $n/2$ bit strings.

8.2 Parameter selection

The parameters L_1 and L_2 and L_H for the hash-function specified in this clause shall satisfy $L_1 = 4n$, $L_2 = 8n$, and L_H is equal to $2n$.

8.3 Padding method

The padding method for use with this hash-function shall be that specified in clause A.3 of ISO/IEC 10118-1:2000, such that $r = n$.

8.4 Initializing value

The selection of the *IV* (of length $8n$) for use with this hash-function is beyond the scope of this part of ISO/IEC 10118. The value of the *IV* shall be agreed upon and fixed by the users of the hash-function.

8.5 Round-function

The round-function ϕ has 8 parallel encryptions, and 8 n-bit chaining variables $H_j^{1\text{-}8}$.

In every iteration, 4 n-bit data blocks, $D_j^{1\text{-}4}$ (of length $L_1 = 4n$ bits) are combined with $H_{j\text{-}1}^{1\text{-}8}$, the previous output of the round-function (of length $L_2 = 8n$ bits), to yield $H_j^{1\text{-}8}$ (of length $L_2 = 8n$ bits).

The round-function is based on a linear mapping γ_1, that takes as input 12 n-bit strings $I^{1\text{-}12}$ and maps them to 8 n-bit strings $X^{1\text{-}8}$ and to 8 n-bit strings $Y^{1\text{-}8}$. The mapping uses 8 $2n$-bit auxiliary strings R^0, R^1, M^0, M^1, ..., M^5. The mapping γ_1 is defined by the following steps:

i) for $i = 0$ to 5 do $\{ M^{iL} := I^{2i+1}$; $M^{iR} := I^{2i+2}$;}

$R^0 := 0$; $R^1 := 0$;

ii) for $i = 0$ to 5 do {

$B := R^1 \oplus M^i$;

$R^1 := R^0 \oplus \beta(B)$;

$R^0 := B$; }

iii) for $i = 1$ to 8 do $\{ X^i := I^i$;}

$Y^1 := R^{0L}$;

$Y^2 := R^{0R}$;

$Y^3 := R^{1L}$;

$Y^4 := R^{1R}$;

for $i = 1$ to 4 do $\{ Y^{4+i} := I^{8+i}$;}

The round-function has the following form ($1 \leq j \leq q$).

$$(X_j^{1\text{-}8}, Y_j^{1\text{-}8}) := \gamma_1(H_{j\text{-}1}^{1\text{-}8}, D_j^{1\text{-}4}) ;$$

for $i = 1$ to 8 do $\{ H_j^i := f_i(X_j^i, Y_j^i)$;}

The round-function is illustrated in Figure 3a and the linear mapping γ_1 in Figure 3b.

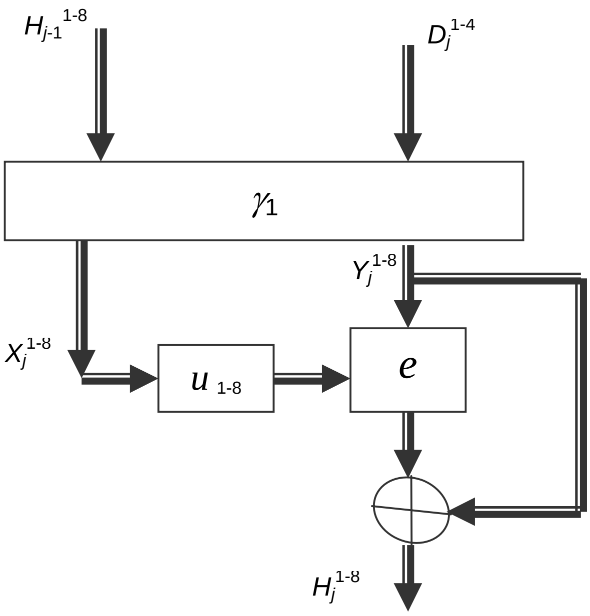

Figure 3a — Round-function of hash-function three

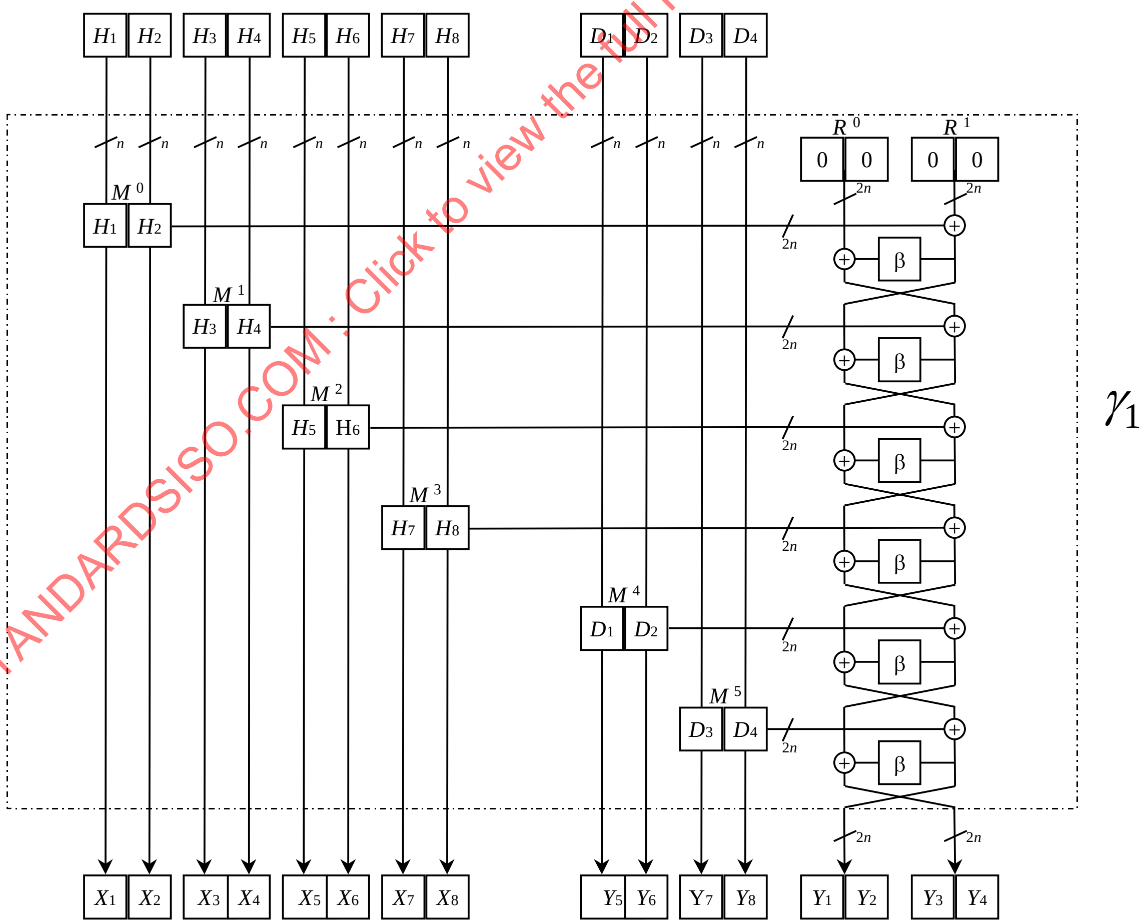

Figure 3b — Linear mapping γ_1 of hash-function three

8.6 Output transformation

After processing of the padded message, the chaining variables have the values $H_q^{1\text{-}8}$. Perform four additional iterations of the round-function with the data inputs

$D_{q+1}^{1\text{-}4} = H_q^{1\text{-}4}$,

$D_{q+2}^{1\text{-}4} = H_q^{5\text{-}8}$,

$D_{q+3}^{1\text{-}4} = H_q^{1\text{-}4}$, and

$D_{q+4}^{1\text{-}4} = H_q^{5\text{-}8}$, respectively.

The output L_H of the hash-function then consists of $H_{q+4}^{1} \,||\, H_{q+4}^{2}$. The output transformation requires 26 encryptions (in the last iteration only two encryptions need to be performed).

9 Hash-function four

The hash-function specified in this clause provides hash-codes of length L_H, where L_H is equal to 3*n* for even values of *n*.

9.1 General

See 8.1 for specific definitions relevant to this hash-function.

9.2 Parameter selection

The parameters L_1 and L_2 and L_H for the hash-function specified in this clause shall satisfy $L_1 = 3n$, $L_2 = 9n$, and L_H is equal to 3*n*.

9.3 Padding method

The padding method for use with this hash-function shall be that specified in clause A.3 of ISO/IEC 10118-1:2000, such that $r = n$.

9.4 Initializing value

The selection of the *IV* (of length 9*n*) for use with this hash-function is beyond the scope of this part of ISO/IEC 10118. The value of the *IV* shall be agreed upon and fixed by the users of the hash-function.

9.5 Round-function

The round-function ϕ has 9 parallel encryptions, and 9 *n*-bit chaining variables $H_j^{1\text{-}9}$.

In every iteration, 3 *n*-bit data blocks, $D_j^{1\text{-}3}$ (of length $L_1 = 3n$ bits) are combined with $H_{j-1}^{1\text{-}9}$, the previous output of the round-function (of length $L_2 = 9n$ bits), to yield $H_j^{1\text{-}9}$ (of length $L_2 = 9n$ bits).

The round-function is based on a linear mapping γ_2, that takes as input 12 *n*-bit strings $I^{1\text{-}12}$ and maps them to 9 *n*-bit strings $X^{1\text{-}9}$ and to 9 *n*-bit strings $Y^{1\text{-}9}$. The mapping uses 9 2*n*-bit auxiliary strings R^0, R^1, R^2, M^0, M^1, …, M^5. The mapping γ_2 is defined by the following steps:

i) for $i = 0$ to 5 do $\{ M^{iL} := I^{2i+1} ; M^{iR} := I^{2i+2} ;\}$

$R^0 := 0 ; R^1 := 0 ; R^2 := 0 ;$

ii) for $i = 0$ to 5 do {

$B := R^2 \oplus M^i$;

$U := \beta(B)$;

$R^2 := R^1 \oplus U$;

$R^1 := R^0 \oplus U$;

$R^0 := B$; }

iii) for $i = 1$ to 9 do { $X^i := I^i$;}

$Y^1 := R^{0L}$;

$Y^2 := R^{0R}$;

$Y^3 := R^{1L}$;

$Y^4 := R^{1R}$;

$Y^5 := R^{2L}$;

$Y^6 := R^{2R}$;

for $i = 1$ to 3 do { $Y^{6+i} := I^{9+i}$;}

The round-function has the following form ($1 \leq j \leq q$).

$(X_j^{1\text{-}9}, Y_j^{1\text{-}9}) := \gamma_2(H_{j\text{-}1}^{1\text{-}9}, D_j^{1\text{-}3})$;

for $i = 1$ to 9 do { $H_j^i := f_i\,(X_j^i, Y_j^i)$;}

The round-function is illustrated in Figure 4a and the linear mapping γ_2 in Figure 4b.

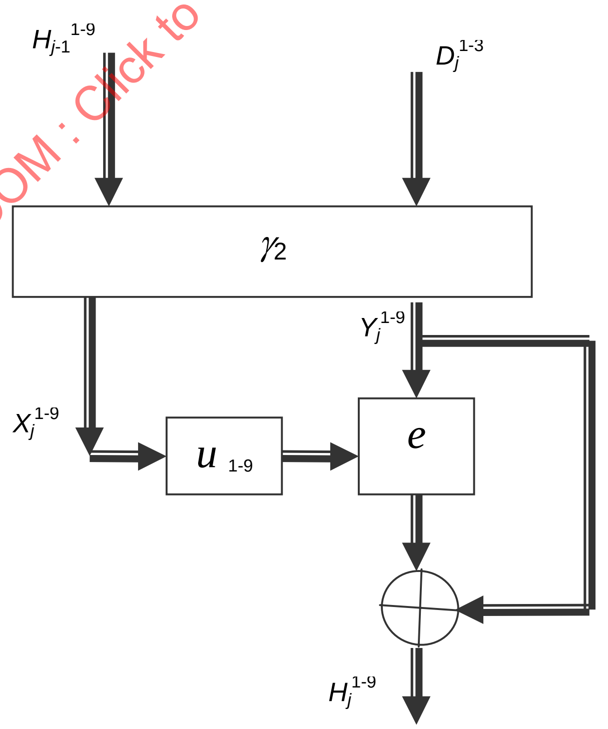

Figure 4a — Round-function of hash-function four

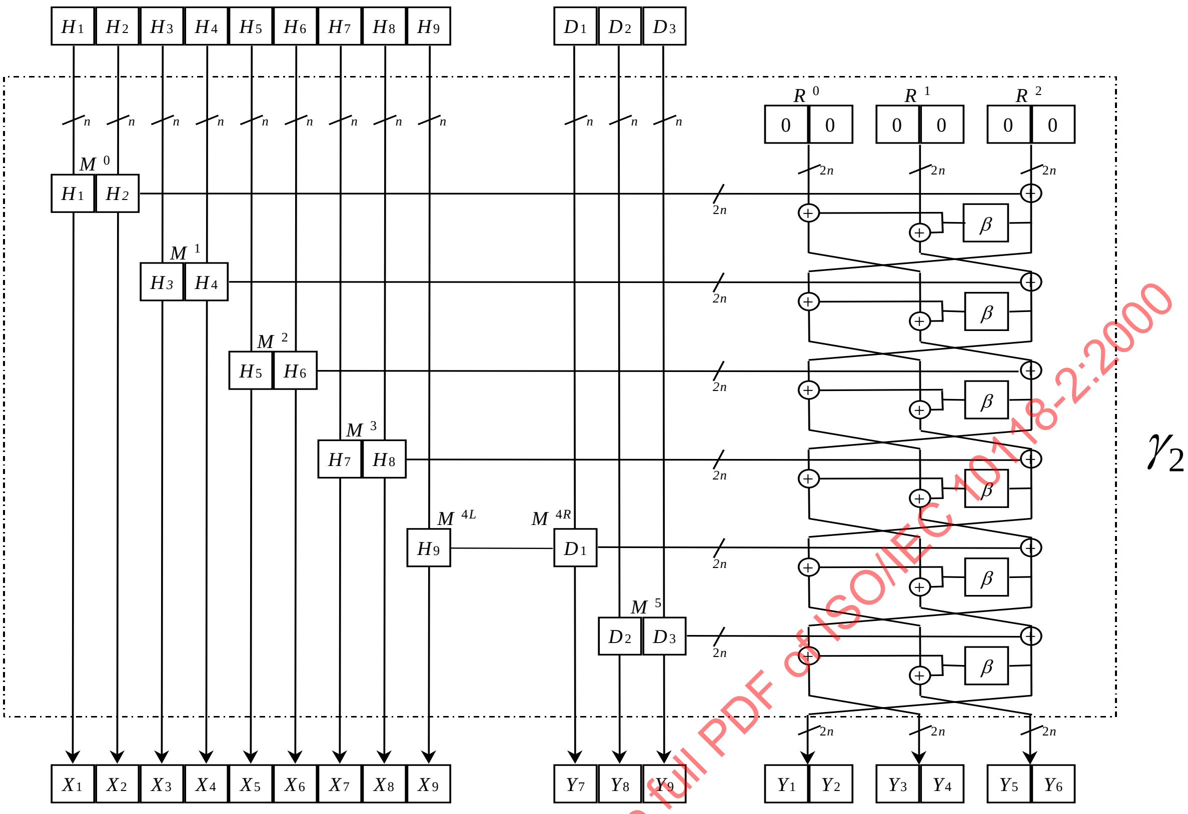

Figure 4b — Linear mapping γ_2 of hash-function four

9.6 Output transformation

After processing of the padded message, the chaining variables have the values $H_q^{1\text{-}9}$. Perform four additional iterations of the round-function with the message inputs

$D_{q+1}^{1\text{-}3} = H_q^{1\text{-}3}$,

$D_{q+2}^{1\text{-}3} = H_q^{4\text{-}6}$,

$D_{q+3}^{1\text{-}3} = H_q^{7\text{-}9}$,

$D_{q+4}^{1\text{-}3} = H_q^{1\text{-}3}$ respectively.

The output of the hash-function then consists of $H_{q+4}^{1} || H_{q+4}^{2} || H_{q+4}^{3}$ The output transformation requires 30 encryptions (in the last iteration only three encryptions need to be performed).

Annex A
(informative)

Use of DEA

A.1 General

This annex presents a way of using the DEA (ANSI X3.92) in conjunction with the hashing operations specified in this part of ISO/IEC 10118. The DEA is also known under the name DES.

The numbering of the bits is as in ANSI X3.92 [2].

These methods have been described in [3]. The parameters for DEA are n = 64 and L_K = 64.

A.2 Hash-function one

See clause 6.

IV should be equal to '5252525252525252' (in hexadecimal notation).

The transformation u should be chosen as follows. Let $X = x_1x_2...x_{64}$ be the binary decomposition of a 64-bit string X. Then $Y = u(X)$ is the string obtained after forcing the bits x_2 and x_3 to the values '10', and replacing the bits x_8, x_{16}, x_{24}, x_{32}, x_{40}, x_{48}, x_{56}, x_{64} with x_8', x_{16}', x_{24}', x_{32}', x_{40}', x_{48}', x_{56}', x_{64}' respectively, where x_{8i}' represents the parity bit for the preceding 7 bits of X, namely, x_{8i-7}, x_{8i-6}, x_{8i-5}, x_{8i-4}, x_{8i-3}, x_{8i-2}, x_{8i-1}. The result is: $Y = x_1'$ 10 x_4 x_5 x_6 x_7 x_8' x_9 x_{10}... x_{63} x_{64}'. The justification for fixing bits 2 and 3 of X is described in [7] with respect to the IBM MDC-2 algorithm; however, the same justification applies to this hash-function.

NOTE It is believed that finding collisions for the round-function and for the hash-function requires 2^{27} DES encryptions.

A.3 Hash-function two

See clause 7.

IV^L should be equal to '5252525252525252' (in hexadecimal notation).

IV^R should be equal to '2525252525252525' (in hexadecimal notation).

The transformation u should be the same as in clause A.2. The transformation u' should be chosen as follows. Let $X = x_1x_2...x_{64}$ be the binary decomposition of a 64-bit string X. Then $Y = u'(X)$ is the string obtained after forcing the bits x_2 and x_3 to the values '01', and replacing the bits x_8, x_{16}, x_{24}, x_{32}, x_{40}, x_{48}, x_{56}, x_{64} with x_8', x_{16}', x_{24}', x_{32}', x_{40}', x_{48}', x_{56}', x_{64}' respectively, where x_{8i}' represents the parity bit for the preceding 7 bits of X, namely, x_{8i-7}, x_{8i-6}, x_{8i-5}, x_{8i-4}, x_{8i-3}, x_{8i-2}, x_{8i-1}. The result is: $Y = x_1'$ 01 x_4 x_5 x_6 x_7 x_8' x_9 x_{10}... x_{63} x_{64}'. The justification for fixing bits 2 and 3 of X is described in [7] with respect to the IBM MDC-2 algorithm; however, the same justification applies to this hash-function.

NOTE It is believed that finding collisions for the round-function and for the hash-function requires 2^{55} DES encryptions.

A.4 Hash-function three

See clause 8.

IV^1, IV^2, ... IV^8 shall be equal to `5252525252525252' (in hexadecimal notation).

The transformations u_1, u_2, ..., u_8 shall be chosen as follows. Let $X = x_1\ x_2\ ...\ x_{64}$ be the binary decomposition of the 64-bit string X. Then $Y = u_i(X)$ is the string obtained after forcing the bits x_1, x_2, … x_5 to the values given in Table A.1, and replacing the bits x_8, x_{16}, x_{24}, x_{32}, x_{40}, x_{48}, x_{56}, x_{64} with x_8', x_{16}', x_{24}', x_{32}', x_{40}', x_{48}', x_{56}', x_{64}' respectively, where x_{8i}' represents the parity bit for the preceding 7 bits of X, namely, x_{8i-7}, x_{8i-6}, x_{8i-5}, x_{8i-4}, x_{8i-3}, x_{8i-2}, x_{8i-1}.

Table A.1 — Hash-function three: Values of key bits No. 1, 2, 3, 4 and 5 in the eight subfunctions

Subfunction *i*	Subfunction *i*
1	00101
2	01001
3	10001
4	00110
5	01010
6	10010
7	01100
8	10100

NOTE It is believed that finding collisions for the round-function and for the hash-function requires 2^{51} DES encryptions.

A.5 Hash-function four

See clause 9.

IV^1, IV^2, ... IV^9 shall be equal to `5252525252525252' (in hexadecimal notation).

The transformations u_1, u_2, ..., u_9 shall be chosen as follows. Let $X = x_1\ x_2\ ...\ x_{64}$ be the binary decomposition of the 64-bit string X. Then $Y = u_i(X)$ is the string obtained after forcing the bits x_1, x_2, … x_5 to the values given in Table A.2, and replacing the bits x_8, x_{16}, x_{24}, x_{32}, x_{40}, x_{48}, x_{56}, x_{64} with x_8', x_{16}', x_{24}', x_{32}', x_{40}', x_{48}', x_{56}', x_{64}' respectively, where x_{8i}' represents the parity bit for the preceding 7 bits of X, namely, x_{8i-7}, x_{8i-6}, x_{8i-5}, x_{8i-4}, x_{8i-3}, x_{8i-2}, x_{8i-1}.

Table A.2 — Hash-function four: Values of key bits No. 1, 2, 3, 4 and 5 in the nine subfunctions

Subfunction *i*	Subfunction *i*
1	00101
2	01001
3	10001
4	00110
5	01010
6	10010
7	01100
8	10100
9	11000

NOTE It is believed that finding collisions for the round-function and for the hash-function requires 2^{76} DES encryptions.

A.6 Motivation

The DEA has some properties that are known to be undesirable, if the algorithm is used in hashing constructions. First of all, there are 4 weak keys, where the encryption function equals the decryption function. In addition, for these 4 weak keys there are 2^{32} fixed points, that is, values of plaintext which encrypt to themselves. Second, there are 16 pairs of semi-weak keys, for which the encryption function induced by one key equals the decryption function of the other key. DES also has the complementation property: if both plaintext and key are complemented, the ciphertext will be complemented as well.

For hash-function one and two, fixing 2 bits of the key as indicated above is a necessary and sufficient condition to avoid weak and semi-weak keys. Hash-function one needs one fixed value and hash-function two needs two fixed values. These values must have the following properties:

— All values must be different.

— None of the values must enable the use of a weak nor a semi-weak key.

For hash-function three and four, fixing 5 bits of the key as indicated above is a necessary and sufficient condition to avoid weak keys, semi-weak keys, and the complementation property.

Hash-function three requires 8 fixed values and hash-function four requires 9 fixed values. These values must have the following properties:

— All values must be different.

— None of the values must enable the use of a weak nor a semi-weak key.

— None of the values is the complemented value of another value.

The fact that the above conditions are met can be derived from the following observation. Consider the 5 key bits in positions 1, 2, 3, 4 and 5. For all weak and semi-weak keys of DEA, these 5 bits take one of the following values: 00000, 11111, 00011, or 11100.

Annex B
(informative)

Examples

B.1 General

This annex gives examples for the computation of a hash-code using the first two hash-functions specified in annex A of this part of ISO/IEC 10118 and the padding methods specified in annex A of ISO/IEC 10118-1:2000.

The data string is the 7-bit ASCII code as described in [8] (no parity) for "Now_is_the_time_for_all_", where "_" denotes a blank, in hexadecimal notation:

'4E6F772069732074686520746696D6520666F7220616C6C20'

B.2 Hash-function one

See A.2.

Padding method 1

j	D_j	H_{j-1}	H_j
1	4E6F772069732074	5252525252525252	858A260F7391482D
2	68652074696D6520	858A260f7391482D	BDE06E66A0454081
3	666F7220616C6C20	BDE06E66A0454081	FF87B67E29BB87B1

Padding method 2

j	D_j	H_{j-1}	H_j
1	4E6F772069732074	5252525252525252	858A260F7391482D
2	68652074696D6520	858A260F7391482D	BDE06E66A0454081
3	666F7220616C6C20	BDE06E66A0454081	FF87B67E29BB87B1
4	8000000000000000	FF87B67E29BB87B1	D992E6CBDFD9BA81

B.3 Hash-function two

See A.3.

Padding method 1

j	D_j	H_{j-1}^{L}	H_{j-1}^{R}
1	4E6F77206973274	5252525252525252	2525252525252525
2	68652074696D6520	858A260FFD4873A8	49771DD37391482D
3	666F7220616C620	B002740352F7CF4F	CFE8087E1B93CCB2