INTERNATIONAL STANDARD

ISO/IEC 9797-2

First edition
2002-06-01

Information technology — Security techniques — Message Authentication Codes (MACs) —

Part 2: Mechanisms using a dedicated hash-function

Technologies de l'information — Techniques de sécurité — Codes d'authentification de message (MAC) —

Partie 2: Mécanismes utilisant une fonction de hachage

Reference number
ISO/IEC 9797-2:2002(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

Foreword

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

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

The main task of the joint technical committee is to prepare International Standards. 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.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 9797 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 9797-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

ISO/IEC 9797 consists of the following parts, under the general title *Information technology — Security techniques — Message Authentication Codes (MACs)*:

— *Part 1: Mechanisms using a block cipher*

— *Part 2: Mechanisms using a dedicated hash-function*

Further parts may follow.

Annexes A and B of this part of ISO/IEC 9797 are for information only.

Introduction

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this part of ISO/IEC 9797 may involve the use of a patent concerning MAC Algorithm 1 (MDx-MAC) given in Clause 6.

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

The holder of this patent right has assured the ISO and IEC that he is willing to negotiate licenses 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 the ISO and IEC. Information may be obtained from:

Entrust Technologies, Technology Licensing Dept., 750 Heron Road, Ottawa, Ontario, Canada K1V 1A7.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 9797 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 — Message Authentication Codes (MACs) —
Part 2:
Mechanisms using a dedicated hash-function

1 Scope

This part of ISO/IEC 9797 specifies three MAC algorithms that use a secret key and a hash-function (or its round-function) with an n-bit result to calculate an m-bit MAC. These mechanisms can be used as data integrity mechanisms to verify that data has not been altered in an unauthorised manner. They can also be used as message authentication mechanisms to provide assurance that a message has been originated by an entity in possession of the secret key. The strength of the data integrity mechanism and message authentication mechanism is dependent on the length (in bits) k and secrecy of the key, on the length (in bits) n of a hash-code produced by the hash-function, on the strength of the hash-function, on the length (in bits) m of the MAC, and on the specific mechanism.

The three mechanisms specified in this part of ISO/IEC 9797 are based on the dedicated hash-functions specified in ISO/IEC 10118-3. The first mechanism specified in this part of ISO/IEC 9797 is commonly known as MDx-MAC. It calls the complete hash-function once, but it makes a small modification to the round-function by adding a key to the additive constants in the round-function. The second mechanism specified in this part of ISO/IEC 9797 is commonly known as HMAC. It calls the complete hash-function twice. The third mechanism specified in this part of ISO/IEC 9797 is a variant of MDx-MAC that takes as input only short strings (at most 256 bits). It offers a higher performance for applications that work with short input strings only.

This part of ISO/IEC 9797 can be applied to the security services of any security architecture, process, or application.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9797. 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 9797 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 646:1991, *Information technology — ISO 7-bit coded character set for information interchange.*

ISO 7498-2:1989, *Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 2: Security Architecture.*

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

ISO/IEC 10118-3:1998, *Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions.*

3 Terms and definitions

3.1 For the purposes of this part of ISO/IEC 9797, the following definitions from ISO/IEC 9797-1 apply.

3.1.1 MAC algorithm key: a key that controls the operation of a MAC algorithm.

3.1.2 Message Authentication Code (MAC): the string of bits which is the output of a MAC algorithm.

NOTE — A MAC is sometimes called a cryptographic check value (see for example ISO 7498-2).

3.1.3 Message Authentication Code (MAC) algorithm: an algorithm for computing a function which maps strings of bits and a secret key to fixed-length strings of bits, satisfying the following two properties:

- for any key and any input string the function can be computed efficiently;
- for any fixed key, and given no prior knowledge of the key, it is computationally infeasible to compute the function value on any new input string, even given knowledge of the set of input strings and corresponding function values, where the value of the ith input string may have been chosen after observing the value of the first $i-1$ function values.

NOTES

1 A MAC algorithm is sometimes called a cryptographic check function (see for example ISO 7498-2).

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

3.1.4 output transformation: a function that is applied at the end of the MAC algorithm, before the truncation operation.

3.2 This part of ISO/IEC 9797 makes use of the following general security-related terms defined in ISO/IEC 10118-1.

3.2.1 collision-resistant hash-function: hash-function satisfying the following property:

- it is computationally infeasible to find any two distinct inputs which map to the same output.

3.2.2 data string (data): string of bits which is the input to a hash-function.

3.2.3 hash-code: string of bits which is the output of a hash-function.

3.2.4 hash-function: function which maps strings of bits to fixed-length strings of bits, satisfying the following two properties:

- for a given output, it is computationally infeasible to find an input which maps to this output;
- for a given input, it is computationally infeasible to find a second input which maps to the same output.

3.2.5 initializing value: value used in defining the starting point of a hash-function.

3.2.6 padding: appending extra bits to a data string.

3.3 This part of ISO/IEC 9797 makes use of the following general security-related terms defined in ISO/IEC 10118-3.

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

3.3.2 round-function: function $\phi(.,.)$ that transforms two binary strings of lengths L_1 and L_2 to a binary string of length L_2.

NOTE — 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.

3.3.3 word: string of 32 bits.

4 Symbols and notation

This part of ISO/IEC 9797 makes use of the following symbols and notation defined in ISO/IEC 9797-1:

D, D' data strings to be input to the MAC algorithm.

m the length (in bits) of the MAC.

q the number of blocks in the data string D after the padding and splitting process.

$j \sim X$ the string obtained from the string X by taking the leftmost j bits of X.

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

$X \| Y$ concatenation of bit-strings X and Y (in that order).

:= a symbol denoting the 'set equal to' operation used in the procedural specifications of MAC algorithms, where it indicates that the value of the string on the left side of the symbol shall be made equal to the value of the expression on the right side of the symbol.

For the purposes of this part of ISO/IEC 9797, the following symbols and notation apply:

$\overline{D}$ padded data string.

h hash-function.

h' hash-function h with modified constants and modified IV.

$\overline{h}$ simplified hash-function h without the padding and length appending.

NOTE — $\overline{h}$ shall only be applied to input strings with a length that is a positive integer multiple of L_1.

H', H'' strings of L_2 bits which are used in the MAC algorithm computation to store an intermediate result.

IV', IV_1, IV_2 initializing values.

k length (in bits) of the MAC algorithm key.

K secret MAC algorithm key.

K', K_0, K_1, K_2, $\overline{K}$, $\overline{K_1}$, $\overline{K_2}$ secret MAC algorithm derived keys.

$\tilde{L}$ the bit string encoding the message length in MAC Algorithm 3.

$OPAD$, $IPAD$ constant strings used in MAC Algorithm 2.

R, S_0, S_1, S_2 constant strings used in the computation of the constants for MAC Algorithm 1 and MAC Algorithm 3.

T_0, T_1, T_2 constant strings used in the key derivation for MAC Algorithm 1 and MAC Algorithm 3.

U_0, U_1, U_2 constant strings used in the key derivation for MAC Algorithm 1 and MAC Algorithm 3.

ϕ' round-function with modified constants.

$K_1[i]$ the ith word of the 128-bit string K_1, i.e.,

$$K_1 = K_1[0] \,\|\, K_1[1] \,\|\, K_1[2] \,\|\, K_1[3]\,.$$

This part of ISO/IEC 9797 makes use of the following symbols and notation defined in ISO/IEC 10118-1:

H hash-code.

IV initializing value.

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

This part of ISO/IEC 9797 makes use of the following symbols and notation defined in ISO/IEC 10118-3:

C_i, C_i' constant words used in the round-functions.

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.

ϕ a round-function, i.e., if X and 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.

$\uplus$ 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.

5 Requirements

Users who wish to employ a MAC algorithm from this part of ISO/IEC 9797 shall select:

- a MAC algorithm from amongst those specified in Clauses 6, 7, and 8;
- a dedicated hash-function from those functions specified in ISO/IEC 10118-3; and
- the length (in bits) m of the MAC.

Agreement on these choices amongst the users is essential for the purpose of the operation of the data integrity mechanism.

For MAC Algorithm 1 and 2, the length m of the MAC shall be a positive integer less than or equal to the length of the hash-code L_H. For MAC Algorithm 3, the length m of the MAC shall be less than or equal to half the length of the hash-code, i.e., $m \leq L_H/2$.

The length in bits of the data string D shall be at most $2^{64} - 1$ for MAC Algorithm 1 and 2, and shall be at most 256 for MAC Algorithm 3.

The selection of a specific MAC algorithm, dedicated hash-function, and value for m are beyond the scope of this part of ISO/IEC 9797.

> NOTE — These choices affect the security level of the MAC algorithm. For a detailed discussion, see Annex B.

The key used for calculating and verifying the MAC shall be the same. If the data string is also being enciphered, the key used for the calculation of the MAC shall be different from that used for encipherment.

> NOTE — It is considered to be good cryptographic practice to have independent keys for confidentiality and for data integrity.

6 MAC Algorithm 1

> NOTE — This clause contains a description of MDx-MAC [5]. More specifically, with Dedicated Hash-Function 1 this mechanism is also known as RIPEMD-160-MAC, with Dedicated Hash-Function 2 this mechanism is also known as RIPEMD-128-MAC, and with Dedicated Hash-Function 3 this mechanism is also known as SHA-1-MAC.

MAC Algorithm 1 requires one application of the hash-function to compute a MAC value, but requires that the constants in the round-function are modified.

The hash-function shall be selected from Dedicated Hash-Function 1, 2 and 3 from ISO/IEC 10118-3:1998.

The key size k in bits shall be at most 128 bits.

6.1 Description of MAC Algorithm 1

MAC algorithm 1 requires the following five steps: key expansion, modification of the constants and the IV, hashing operation, output transformation, and truncation.

6.1.1 Step 1 (key expansion)

If K is shorter than 128 bits, concatenate K to itself a sufficient number of times and select the leftmost 128 bits to form the 128-bit key K' (if the length (in bits) of K is equal to 128, $K' := K$):

$$K' := 128 \sim (K \,\|\, K \,\|\, \ldots \,\|\, K)\,.$$

Compute the subkeys K_0, K_1, and K_2 as follows:

$$\begin{aligned} K_0 &:= \overline{h}(K' \,\|\, U_0 \,\|\, K') \\ K_1 &:= 128 \sim \overline{h}(K' \,\|\, U_1 \,\|\, K') \\ K_2 &:= 128 \sim \overline{h}(K' \,\|\, U_2 \,\|\, K')\,. \end{aligned}$$

Here U_0, U_1, and U_2 are 768-bit constants that are defined in Clause 6.3, and $\overline{h}$ denotes the simplified hash-function h, i.e., without the padding and length appending.

> NOTE — Padding and length appending can be omitted because in this case the length of the input string is always $2L_1$ bits.

The derived key K_1 is split into four words denoted $K_1[i]$ $(0 \le i \le 3)$, i.e.

$$K_1 = K_1[0] \,\|\, K_1[1] \,\|\, K_1[2] \,\|\, K_1[3]\,.$$

For the conversion of a string into words, a byte ordering convention is required. The byte ordering convention for this conversion is that which is defined for each dedicated hash-function in ISO/IEC 10118-3.

6.1.2 Step 2 (modification of the constants and the IV)

The additive constants used in the round-function are modified by the addition mod 2^{32} of one of the four words of K_1, e.g.,

$$C_0 := C_0 \uplus K_1[0]\,.$$

Clause 6.3 indicates which word of K_1 is added to each constant. The initial value IV of the hash function is replaced by $IV' := K_0$. The resulting hash-function is denoted by h', and its round-function is denoted by ϕ'.

6.1.3 Step 3 (hashing operation)

The string which is input to the modified hash-function h' is equal to the data string D, i.e.

$$H' := h'(D)\,.$$

6.1.4 Step 4 (output transformation)

The modified round-function ϕ' is applied one more time, with as first input the string $K_2 \,\|\, (K_2 \oplus T_0) \,\|\, (K_2 \oplus T_1) \,\|\, (K_2 \oplus T_2)$, and as second input the string H' (the result of Step 3) i.e.

$$H'' := \phi'(K_2 \,\|\, (K_2 \oplus T_0) \,\|\, (K_2 \oplus T_1) \,\|\, (K_2 \oplus T_2), H')\,.$$

Here T_0, T_1, and T_2 are 128-bit strings defined in Clause 6.3 for each dedicated hash-function.

> NOTE — The output transformation corresponds to processing an additional data block derived from K_2 *after* padding and appending of the length field.

6.1.5 Step 5 (truncation)

The MAC of m bits is derived by taking the leftmost m bits of the string H'', i.e.

$$\text{MAC} := m \sim H''\,.$$

6.2 Efficiency

Assume that the padded data string (here the padding algorithm is defined for a specific hash-function) contains q blocks; then MAC Algorithm 1 requires $q + 7$ applications of the round-function.

This can be reduced to $q + 1$ applications of the round-function by pre-computing the values K_0, K_1, and K_2. and by replacing the initial value IV by IV' in the application of the hash-function.

It is recommended to make this modification to the code of the hash-function together with the mandatory modification required for Step 2.

For long input strings, MAC Algorithm 1 has a performance which is comparable to that of the hash-function used.

6.3 Computation of the constants

The constants described in this clause will be used in both MAC Algorithm 1 and MAC Algorithm 3 specified in Clause 8.

The strings T_i and U_i are fixed elements of the description of the MAC algorithm. They are computed (only once) using the hash-function; they are different for each of the three hash-functions.

The 128-bit constants T_i and 768-bit constants U_i are defined as follows. The definition of T_i involves the 496-bit constant $R =$"ab...yzAB...YZ01...89" and 16-bit constants S_0, S_1, S_2, where S_i is the 16-bit string formed by repeating twice the 8-bit representation of i (e.g., the hexadecimal representation of S_1 is 3131). In both cases ASCII coding is used; this is equivalent to coding using ISO/IEC 646:1991.

$$\begin{aligned} &\textit{for } i := 0 \textit{ to } 2 \quad T_i \quad := 128 \sim \overline{h}(S_i \,\|\, R) \\ &\textit{for } i := 0 \textit{ to } 2 \quad U_i \quad := T_i \,\|\, T_{i+1} \,\|\, T_{i+2} \,\|\, T_i \,\|\, T_{i+1} \,\|\, T_{i+2} \end{aligned}$$

where the subscripts in T_i are taken modulo 3.

For all constants C_i, C'_i and all words $K_1[i]$ the most significant bit corresponds to the left-most bit. The constants C_i and C'_i are presented in hexadecimal representation.

6.3.1 Dedicated Hash-Function 1

The 128-bit constant strings T_i for Dedicated Hash-Function 1 are defined as follows (in hexadecimal representation):

$$\begin{aligned} T_0 &= \texttt{1CC7086A046AFA22353AE88F3D3DACEB} \\ T_1 &= \texttt{E3FA02710E491D851151CC34E4718D41} \\ T_2 &= \texttt{93987557C07B8102BA592949EB638F37} \end{aligned}$$

Two sequences of constant words $C_0, C_1, \ldots, C_{79}$ and $C'_0, C'_1, \ldots, C'_{79}$ are used in the round-function of Dedicated Hash-Function 1. They are defined as follows:

$$\begin{aligned} C_i &= K_1[0] \uplus \texttt{00000000}, \quad (0 \le i \le 15), \\ C_i &= K_1[1] \uplus \texttt{5A827999}, \quad (16 \le i \le 31), \\ C_i &= K_1[2] \uplus \texttt{6ED9EBA1}, \quad (32 \le i \le 47), \\ C_i &= K_1[3] \uplus \texttt{8F1BBCDC}, \quad (48 \le i \le 63), \\ C_i &= K_1[0] \uplus \texttt{A953FD4E}, \quad (64 \le i \le 79), \end{aligned}$$

$$\begin{aligned} C'_i &= K_1[1] \uplus \texttt{50A28BE6}, \quad (0 \le i \le 15), \\ C'_i &= K_1[2] \uplus \texttt{5C4DD124}, \quad (16 \le i \le 31), \\ C'_i &= K_1[3] \uplus \texttt{6D703EF3}, \quad (32 \le i \le 47), \\ C'_i &= K_1[0] \uplus \texttt{7A6D76E9}, \quad (48 \le i \le 63), \\ C'_i &= K_1[1] \uplus \texttt{00000000}, \quad (64 \le i \le 79). \end{aligned}$$

6.3.2 Dedicated Hash-Function 2

The 128-bit constant strings T_i for Dedicated Hash-Function 2 are defined as follows (in hexadecimal representation):

$$\begin{aligned} T_0 &= \texttt{FD7EC18964C36D53FC18C31B72112AAC} \\ T_1 &= \texttt{2538B78EC0E273949EE4C4457A77525C} \\ T_2 &= \texttt{F5C93ED85BD65F609A7EB182A85BA181} \end{aligned}$$

Two sequences of constant words $C_0, C_1, \ldots, C_{63}$ and $C'_0, C'_1, \ldots, C'_{63}$ are used in the round-function of Dedicated Hash-Function 2. They are defined as follows:

$$\begin{aligned} C_i &= K_1[0] \uplus \texttt{00000000}, \quad (0 \le i \le 15), \\ C_i &= K_1[1] \uplus \texttt{5A827999}, \quad (16 \le i \le 31), \\ C_i &= K_1[2] \uplus \texttt{6ED9EBA1}, \quad (32 \le i \le 47), \\ C_i &= K_1[3] \uplus \texttt{8F1BBCDC}, \quad (48 \le i \le 63), \end{aligned}$$

$$\begin{aligned} C'_i &= K_1[0] \uplus \texttt{50A28BE6}, \quad (0 \le i \le 15), \\ C'_i &= K_1[1] \uplus \texttt{5C4DD124}, \quad (16 \le i \le 31), \\ C'_i &= K_1[2] \uplus \texttt{6D703EF3}, \quad (32 \le i \le 47), \\ C'_i &= K_1[3] \uplus \texttt{00000000}, \quad (48 \le i \le 63). \end{aligned}$$

6.3.3 Dedicated Hash-Function 3

The 128-bit constant strings T_i for Dedicated Hash-Function 3 are defined as follows (in hexadecimal representation):

$$\begin{aligned} T_0 &= \texttt{1D4CA39FA40417E2AE5A77B49067BBCC} \\ T_1 &= \texttt{9318AFEF5D5A5B46EFCA6BEC0E138940} \\ T_2 &= \texttt{4544209656E14F97005DAC76868E97A3} \end{aligned}$$

A sequence of constant words $C_0, C_1, \ldots, C_{79}$ is used in round-function of Dedicated Hash-Function 3. They are defined as follows:

$$\begin{aligned} C_i &= K_1[0] \uplus \texttt{5A827999}, \quad (0 \le i \le 19), \\ C_i &= K_1[1] \uplus \texttt{6ED9EBA1}, \quad (20 \le i \le 39), \\ C_i &= K_1[2] \uplus \texttt{8F1BBCDC}, \quad (40 \le i \le 59), \\ C_i &= K_1[3] \uplus \texttt{CA62C1D6}, \quad (60 \le i \le 79). \end{aligned}$$

7 MAC Algorithm 2

NOTE — This clause contains a description of HMAC [3].

MAC Algorithm 2 requires two applications of a hash-function to compute a MAC value.

The hash-function shall be selected from ISO/IEC 10118-3, with the requirement that L_1 is a positive integer multiple of 8 and that $L_2 \le L_1$.

NOTE — Dedicated hash-functions 1, 2, and 3 in ISO/IEC 10118-3:1998 satisfy these conditions.

The key size k in bits shall be at least L_2, where L_2 is the size of the hash-code in bits, and at most L_1 bits, where L_1 is the size of the data input of the round-function in bits, i.e., $L_2 \le k \le L_1$.

7.1 Description of MAC Algorithm 2

MAC algorithm 2 requires the following four steps: key expansion, hashing operation, output transformation, and truncation.

7.1.1 Step 1 (key expansion)

Append $(L_1 - k)$ zero bits to the right of the key K; the resulting string of length L_1 is denoted by $\overline{K}$.

The key $\overline{K}$ is expanded to two subkeys $\overline{K}_1$ and $\overline{K}_2$:

- Define the string $IPAD$ as the concatenation of $L_1/8$ times the hexadecimal value '36' (or the binary value '00110110'). Then compute the value $\overline{K_1}$ as the exclusive-or of $\overline{K}$ and the string $IPAD$, i.e.
$$\overline{K_1} := \overline{K} \oplus IPAD\,.$$
- Define the string $OPAD$ as the concatenation of $L_1/8$ times the hexadecimal value '5C' (or the binary value '01011100'). Then compute the value $\overline{K_2}$ as the exclusive-or of $\overline{K}$ and the string $OPAD$, i.e.
$$\overline{K_2} := \overline{K} \oplus OPAD\,.$$

7.1.2 Step 2 (hashing operation)

The string which is input to the hash-function is equal to the concatenation of $\overline{K_1}$ and D, i.e.

$$H' := h(\overline{K_1} \,\|\, D)\,.$$

7.1.3 Step 3 (output transformation)

The string which is input to the hash-function is equal to the concatenation of $\overline{K_2}$ and H', i.e.

$$H'' := h(\overline{K_2} \,\|\, H')\,.$$

7.1.4 Step 4 (truncation)

The MAC of m bits is derived by taking the leftmost m bits of the string H'', i.e.

$$\text{MAC} := m \sim H''\,.$$

7.2 Efficiency

Assume that the padded data string (here the padding algorithm is defined for a specific hash-function) contains q blocks; then MAC Algorithm 2 requires $q + 3$ applications of the round-function.

This can be reduced to $q + 1$ applications of the round-function by modifying the code for the hash-function. One can pre-compute the values $IV_1 := \phi(\overline{K_1}, IV)$ and $IV_2 := \phi(\overline{K_2}, IV)$ and replace the initial value IV by IV_1 in the first application of the hash-function, and by IV_2 in the output transformation (the second application of the hash-function). This also requires a modification of the padding method: indeed, the actual input to the hash-function is now L_1 bits shorter; this means that the value L_1 has to be added to the value L_D.

For long input strings, MAC Algorithm 2 has a performance comparable to that of the hash-function used.

8 MAC Algorithm 3

NOTE — This clause contains a variant of MAC Algorithm 1 that is optimized for short inputs (at most 256 bits).

MAC Algorithm 3 requires seven applications of the simplified round-function to compute a MAC value, but this can be reduced to a single application of this round-function by some pre-computation.

The hash-function shall be selected from Dedicated Hash-Function 1, 2 and 3 from ISO/IEC 10118-3:1998.

The key size k in bits shall be at most 128 bits, and the MAC length m in bits shall be at most $L_H/2$.

8.1 Description of MAC Algorithm 3

MAC algorithm 3 requires the following five steps: key expansion, modification of the constants of the round-function, padding, application of the round-function, and truncation.

8.1.1 Step 1 (key expansion)

If K is shorter than 128 bits, concatenate K to itself a sufficient number of times and select the leftmost 128 bits to form the 128-bit key K' (if the length (in bits) of K is equal to 128, $K' := K$):

$$K' := 128 \sim (K \,\|\, K \,\|\, \ldots \,\|\, K)\,.$$

Compute the subkeys K_0, K_1, and K_2 as follows:

$$\begin{aligned} K_0 &:= \overline{h}(K' \,\|\, U_0 \,\|\, K') \\ K_1 &:= 128 \sim \overline{h}(K' \,\|\, U_1 \,\|\, K') \\ K_2 &:= 128 \sim \overline{h}(K' \,\|\, U_2 \,\|\, K')\,. \end{aligned}$$

Here U_0, U_1 and U_2 are 768-bit constants that are defined in Clause 6.3, and $\overline{h}$ denotes the hash-function h without the padding and length appending.

NOTE — Padding and length appending can be omitted because in this case the length of the input string is always $2L_1$ bits.

The derived key K_1 is split into four words denoted $K_1[i]$ $(0 \leq i \leq 3)$, i.e.

$$K_1 = K_1[0] \,\|\, K_1[1] \,\|\, K_1[2] \,\|\, K_1[3] \,.$$

For the conversion of a string into words, a byte ordering convention is required. The byte ordering convention for this conversion is that which is defined for each dedicated hash-function in ISO/IEC 10118-3.

8.1.2 Step 2 (modification of the constants and the IV)

The additive constants used in the round-function are modified by the addition mod 2^{32} of one of the four words of K_1, e.g.,

$$C_0 := C_0 \uplus K_1[0] \,.$$

Clause 6.3 indicates which word of K_1 is added to each constant. The initial value IV of the hash function is replaced by $IV' := K_0$. The resulting round-function is denoted by ϕ'.

8.1.3 Step 3 (padding)

The padding bits that are added to the original data string are only used for calculating the MAC. Consequently, these padding bits (if any) need not be stored or transmitted with the data. The verifier shall know whether or not the padding bits have been stored or transmitted.

The data string D to be input to the MAC algorithm shall be right-padded with as few (possibly none) '0' bits as necessary to obtain a data string $\overline{D}$ of length 256 bits.

> NOTE — If the data string is empty, the padded data string $\overline{D}$ consists of 256 '0' bits.

8.1.4 Step 4 (application of the round-function)

The bit string $\tilde{L}$ is computed as the binary representation of the length (in bits) L_D of the data string D, left-padded with as few '0' bits as necessary to obtain a 128-bit string. The right-most bit of the bit string $\tilde{L}$ corresponds to the least significant bit of the binary representation of L_D.

The string which is input to the round-function ϕ' (with modified constants) is equal to the concatenation of K_2, $\overline{D}$, and the exclusive-or of K_2 and $\tilde{L}$:

$$H' := \phi'(K_2 \,\|\, \overline{D} \,\|\, (K_2 \oplus \tilde{L}), IV') \,.$$

8.1.5 Step 5 (truncation)

The MAC of m bits is derived by taking the leftmost m bits of the string H', i.e.

$$\text{MAC} := m \sim H' \,.$$

8.2 Efficiency

MAC Algorithm 3 requires seven applications of the round-function.

This can be reduced to a single application of the round-function by pre-computing the values K_0, K_1, and K_2.

Annex A
(informative)

Examples

Table 1 — Input strings for the test values

no.	input string
1	“” (empty string)
2	“a”
3	“abc”
4	“message digest”
5	“abcdefghijklmnopqrstuvwxyz”
6	“abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq”
7	“ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789”
8	“1234567890” repeated 8 times to give 80 characters
9	“a” repeated 1,000,000 times to give 1 million characters

A.1 General

This annex gives examples for the computation of MAC Algorithms 1 and 2 using Dedicated Hash-Functions 1, 2, and 3 from ISO/IEC 10118-3:1998. Nine examples of hash-code calculation are given for each of the hash-functions. The input strings numbered 1 to 9 are contained in Table 1. Throughout this annex we refer to ASCII coding of data strings; this is equivalent to coding using ISO/IEC 646.

The two key values used are the following 128-bit string: key 1 = `00112233445566778899AABBCCDDEEFF` and key 2 = `0123456789ABCDEFFEDCBA9876543210`.

A.2 MAC Algorithm 1

For the examples in this section, the value $m = L_2/2$ has been selected; that is, $m = 80$ for Dedicated Hash-Function 1 and 3 and $m = 64$ for Dedicated Hash-Function 2.

A.2.1 Dedicated Hash-Function 1

key 1: `00112233445566778899AABBCCDDEEFF`	
no.	80-bit MAC result: leftmost 80 bits of
1	`B7F4508111EB8C3B5229C6AED406DE9ECA640133`
2	`BC78F55933BCEB1EE85A906F9E18374F23E310F9`
3	`6300DC20E97A5AA29DB9C7D607D23D126FA36863`
4	`3A2AC89B78EEAB8759F5112BCAD4CD405EEB5D35`
5	`16DC174925BBC27E0C93D426C346846F97F8BC69`
6	`E062210BA5C9C94737BF3A6E85B3B5664FBD1D4E`
7	`9B462D5CBDAE1485FFE10BC001EF9E3AF6D128B5`
8	`88E73A01A1DE36C92D6F9E41F7278D407B4A4CCD`
9	`E7B128E4A1842B750F1E61A486C867C4887A4B21`

key 2: `0123456789ABCDEFFEDCBA9876543210`	
no.	80-bit MAC result: leftmost 80 bits of
1	`B45D6CA84CFB9020E0D5ABA2A7609D3D81F3F57F`
2	`8844375992037D1BCD0D118EE548D70C3F19CBBB`
3	`917C59B8AC7FC19DC25BEF82766412FA16BBC6A7`
4	`E0737CC7976D8F424390CB8798D623D751AFE15A`
5	`D57FAE836870718EFA4BD4A5F2F322A179A8735E`
6	`42B20D4C8FD5E8672760CF83C0478D7BF8021404`
7	`42B20D4C8FD5E8672760CF83C0478D7BF8021404`
8	`10441DF4F68CE8815818DC0FB370ABF87BCA4464`
9	`E06AD21D2AF04DD4217AB03B1A578F036997D01A`

A.2.2 Dedicated Hash-Function 2

key 1: 00112233445566778899AABBCCDDEEFF	
no.	64-bit MAC result: leftmost 64 bits of
1	A47A64E9EDE0741B3FDDE33E5C1C6D78
2	51355051852FDC79FB228EAC905633AD
3	D83940DAFFBD4CBBE6BA30A6F9E63F5F
4	1A7CFE2BB26E973E213C1CB96FA4C2EF
5	798AEAC6046B31907C197BD68E59D376
6	0B8E1D4A571F32657189E22A1F2F4A53
7	B814730F482300C6E474FD255A66D680
8	9060A30758EBE3368D939AC168F1A9FD
9	20763FDEDF01E56FF5756954302C7DE0

key 2: 0123456789ABCDEFFEDCBA9876543210	
no.	64-bit MAC result: leftmost 64 bits of
1	35FA3AC39F50F2A4E3FFC7AF5776B4EB
2	A89E25E6796747B630A2A00B802EA53E
3	66339027A36608EBD932DD551616E7B2
4	1F8779BAD84B50373931211A2761EAD3
5	31BF5B5B7ABAC2567DC0E02F1C3A25D7
6	B5B8BA3B8EA895FBC83CB7588FBD2656
7	8D27BBEC257C848D5CF375EB5EDA4CC7
8	B40B5BF6727DE90B26F770850F059C89
9	76C7BC831B0BCE593DFD44E8E054A373

A.2.3 Dedicated Hash-Function 3

key 1: 00112233445566778899AABBCCDDEEFF	
no.	80-bit MAC result: leftmost 80 bits of
1	C8A8B3C75E6CE7C6C4F79CC19853CCD54ABCB079
2	8DD9AE643BF10BBB7B978EF13EE6C0F480618FB0
3	A738B26A8BD318184E76707A99CAE14C670B9711
4	1EBFE413E55D6B288A2BD01D294A21FD8D4B20BF
5	0CE7BF40A73D977AB4999CF3A9BD1C5BEDC442E9
6	12A6823CC181294F95109073A6AA0C8961B14386
7	9369EE4A043AF1CA6E078D0B8A9CE5C1545440BA
8	B00D37D70A84B762FC0A8A9BC1B15F0E517B5EDF
9	DDDF44613E8559D12C150D022D5FE33F9E0FBACE

key 2: 0123456789ABCDEFFEDCBA9876543210	
no.	80-bit MAC result: leftmost 80 bits of
1	C3A5ECD1E715C7272CFE78BC278086587B040422
2	D5D50FFA7EFDF1B17E96E2EC14DBC4412F7B771F
3	01BFDD568008D412158F5B0C90AE2730DCFB77FB
4	9982E0EE91DB89AE7E7618AD1D649BA43406DBDD
5	ACD04E1004FCE53DECA9EE7AB95DAF97B7C44AA8
6	FADF62DCE789E86E60756AA819EF62C3E5C25E94
7	46DB9A49FB4976D007B14B1574843D019CA99445
8	4EF5BED3E816C530B23F491583C038596BB76FDB
9	BAC6BE6BE6153FECE2891F9DA03824DD4D535D19

A.3 MAC Algorithm 2

For the examples in this section, the value $m = 80$ has been selected.

A.3.1 Dedicated Hash-Function 1

key 1: 00112233445566778899AABBCCDDEEFF	
no.	80-bit MAC result: leftmost 80 bits of
1	9EBEA41FBC24CD80BF2ECFD5B8C8CC8181D3FCAE
2	75CB722C50024C0E8A7A0DBA7D5C36B86D9D1DD5
3	5B48C1749DDED71EDFE0ADE2B944E808E4A65820
4	F9033064567F541235C3944EE95CB476055985D1
5	B37885405B71E025AF0CB574021A562A62733628
6	5C6429B982C8054B5B3348A0D7D2CE24D7032BC1
7	B0A4A451D0926855E52428E16D1FEAA241C4DD9B
8	1CCEEC5122F08A76EBCD8E3DE88610D942D8A5F6
9	45D61908BFF6039E6DE3C037FDCE6191F19F6410

key 2: 0123456789ABCDEFFEDCBA9876543210	
no.	80-bit MAC result: leftmost 80 bits of
1	2FDE5DAF7050D14E6D7ACD2254D17FA3A8CBFCDD
2	239C4020610429A8662BF81A2CAAEA47F8EA0A44
3	89EFFB9F5A6BCEAE3C65D0C9803F3464E5E9E349
4	F5FC87FD5702F5D4E7BB634DA4CB4B41CD505B6C
5	5686C00F69E6C868732C67402AA107CEAB513439
6	525EC4893A221EFD9B6DD351059B40C05B4CE2D3
7	B975ED3893FC8D535376EF49211E2E6B1BB30B90
8	BC201FFA581357C271DAE25104167F3DCC97BADC
9	95A875A1D64D55E677D8E4455E1445E7E940F758

A.3.2 Dedicated Hash-Function 2

key 1: 00112233445566778899AABBCCDDEEFF	
no.	64-bit MAC result: leftmost 64 bits of
1	AD9DB2C1E22AF9AB5CA9DBE5A86F67DC
2	3BF448C762DE00BCFA0310B11C0BDE4C
3	F34EC0945F02B70B8603F89E1CE4C78C
4	E8503A8AEC2289D82AA0D8D445A06BDD
5	EE880B735CE3126065DE1699CC136199
6	794DAF2E3BDEEA2538638A5CED154434
7	3A06EEF165B23625247800BE23E232B6
8	9A4F0159C0952DA43A8D466D46B0AF58
9	19B1B3AF333B894DD86D09427116D0AD

key 2: 0123456789ABCDEFFEDCBA9876543210	
no.	64-bit MAC result: leftmost 64 bits of
1	8931EEEE56A6B257FD1AB5418183D826
2	DBBCF169EA7419D5BA7BD8EB3673FF2D
3	2C4CD07D3162D6A0E338004D6B6FBC9A
4	75BFB25888F4BB77C77AE83AD0817447
5	B1B5DC0FCB7258758855DD1840FCDCE4
6	670D0F7A697B18F1A8AB7D2A2A00DBC1
7	54E315FDB34A61C0475392E5C7852998
8	AD04354D8AA2A623E72E3594EE3535C0
9	6F9B1C0FC06753618D6DB4B007733795

A.3.3 Dedicated Hash-Function 3

key 1: 00112233445566778899AABBCCDDEEFF	
no.	80-bit MAC result: leftmost 80 bits of
1	86C2962E58B3498A2608935AF7726311F2BFB538
2	0497FF21DAE3251DA0ED2F47F5A3B74ABA6B2560
3	6EE2A25F943E3F3EC05225FBB86BA73E2E5D51D2
4	CD4C0D1328DC4A8DC2801001B129AEFC6E0CF9CE
5	89ECE303FAD1E4313950CC3B008CB239B5B85844
6	9DF741057D075D3C4E1533E38A5FF469647194B4
7	188A58390A6EF9827035B81CDF1B5069211F0EE5
8	98A98D6A81FD361030856D2C19742AD8DBC468E7
9	D2986310BA18A78786534882F9C6BCBF06CCE9E3

key 2: 0123456789ABCDEFFEDCBA9876543210	
no.	80-bit MAC result: leftmost 80 bits of
1	2739B6BE63F539EB70FE250346F6382A2DFA345F
2	A0C2711A6B1DA4CD8F85EF1E6FF7BF70B412B477
3	18F570E864FF903D2773D53C2E114E1A62152953
4	A80845A89BA15E941A2457084BC431F3E47759E1
5	14143EA1057B02D20C0157216190A006E30F3D41
6	DAB4B41BA639B4715889406FE18E0C037017E063
7	AEAEA5415B4F266CB15CBEB844E56AEC2DABAD6D
8	3DBA11471EB4FCCF21BAEB0BFF7E20150132C6CF
9	3BB917B8BD8560E89FF9054FBE096CBACA109D5F

A.4 MAC Algorithm 3

For the examples in this section, the value $m = L_2/2$ has been selected, that is, $m = 80$ for Dedicated Hash-Function 1 and 3 and $m = 64$ for Dedicated Hash-Function 2.

A.4.1 Dedicated Hash-Function 1

key 1: 00112233445566778899AABBCCDDEEFF	
no.	80-bit MAC result: leftmost 80 bits of
1	6606EF2D3BBD010F516C65372C3CF0ACF111B3F7
2	F0BC0C81307E17A71F4C40AE0B2AC39FCB23CE12
3	7720FD23925B854F963E8812573CD86EBA61EB66
4	2683D6CE053BA0420E76130EAE2367734B7D2D53
5	DE532D156CBE12464BB6147E99470C471D91F1C6

key 2: 0123456789ABCDEFFEDCBA9876543210	
no.	80-bit MAC result: leftmost 80 bits of
1	4BD390E9EC460AD4866CCB32D091AFBF73E5B6DA
2	CD2847BAB4636C9BCEADCF3D187122A9199DA670
3	15C3910C42638E5EE6DEBD506BD8C4DB94713A3A
4	04148DCB47728E3E57B836A66043D5145879796E
5	829A24010704DBD0EE34A6D607F7B34829E04E95

A.4.2 Dedicated Hash-Function 2

key 1: 00112233445566778899AABBCCDDEEFF	
no.	64-bit MAC result: leftmost 64 bits of
1	aeb2c45f13c0c6f5f10be2f1e3e9c322
2	16874d0e17e4f1c290dd749cceff7834
3	a289aa06aeb8fc99b989c377baadd9d4
4	0d80db68bbf99442dc3d6b83d038def3
5	11dc4a6bd375c64f78bb78ad265ed7cd

key 2: 0123456789ABCDEFFEDCBA9876543210	
no.	64-bit MAC result: leftmost 64 bits of
1	7248481816b8d3af29f5c002ff769ea5
2	dfe1e36ce9792476e0889f1becef18a9
3	9b4f1d21320f4a327f023947554bfc3b
4	3d2d658d0196e4339f42ada50dfcfa6f
5	0a34452d9da70c70183dffddb8eec056

A.4.3 Dedicated Hash-Function 3

key 1: 00112233445566778899AABBCCDDEEFF	
no.	80-bit MAC result: leftmost 80 bits of
1	708F4A226CDE708820643CBEEFDCBE6CB36FB269
2	EAB87BE709D1E5CB62C7489C2B1407130B772760
3	C1BD6F9C908132FEF5187CBE681B42A8C785FBF6
4	F34DEB241D46C6448D67ACE6B8CD4DF00DA23EBC
5	669DED2BD6A1AE0BCFF7D3B74494C1D8161FA0D8

key 2: 0123456789abcdeffedcba9876543210	
no.	80-bit MAC result: leftmost 80 bits of
1	EAF6F9DDBAFD299320FF0FC8E02E2BE62879F341
2	2AAE9DE0A555E7CD7383C27506A467B8DF4E3A33
3	FE6031710329D12090F73F55CFFCC6215F9BEAE9
4	0CCDD9DAB6B0126800EC1CC7A02656E12EDEA42C
5	ABDBC8AAAE4A8CE734432188740A149BDF2D215F