INTERNATIONAL STANDARD

ISO/IEC 9797-1

Second edition
2011-03-01

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

Part 1: Mechanisms using a block cipher

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

Partie 1: Mécanismes utilisant un chiffrement par blocs

Reference number
ISO/IEC 9797-1:2011(E)

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

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members 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 2.

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 document 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 9797-1 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 9797-1:1999), which has been technically revised. MAC Algorithms 5 and 6 of ISO/IEC 9797-1:1999, which consisted of two single CBC-MAC computations, have been replaced by two other MAC algorithms, which perform single CBC-MAC computations and which offer improved efficiency. Annex A on object identifiers has been added. The security analysis in Annex C has been updated and Annex D on the relationship to previous standards has been added.

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*

— *Part 3: Mechanisms using a universal hash-function*

Further parts may follow.

Introduction

In an IT environment, it is often required that one can verify that electronic data has not been altered in an unauthorized manner and that one can provide assurance that a message has been originated by an entity in possession of the secret key. A MAC (Message Authentication Code) algorithm is a commonly used data integrity mechanism that can satisfy these requirements.

This part of ISO/IEC 9797 specifies six MAC algorithms that are based on an n-bit block cipher. They compute a short string as a function of a secret key and a message of variable length.

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 block length (in bits) n and strength of the block cipher, on the length (in bits) m of the MAC, and on the specific mechanism.

The first mechanism specified in this part of ISO/IEC 9797 is commonly known as CBC-MAC (CBC is an abbreviation of Cipher Block Chaining).

The other five mechanisms are variants of CBC-MAC. MAC Algorithms 2, 3, 5 and 6 apply a special transformation at the end of the processing. MAC Algorithm 6 is an optimized variant of MAC Algorithm 2. MAC Algorithm 5 uses the minimum number of encryptions. MAC Algorithm 5 requires only a single block cipher key setup but it needs a longer internal key. MAC Algorithm 4 applies a special transformation at both the beginning and the end of the processing; this algorithm is recommended for use in applications which require that the key length of the MAC algorithm be twice that of the block cipher.

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

Part 1:
Mechanisms using a block cipher

1 Scope

This part of ISO/IEC 9797 specifies six MAC algorithms that use a secret key and an n-bit block cipher to calculate an m-bit MAC.

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

Key management mechanisms are outside the scope of this part of ISO/IEC 9797.

This part of ISO/IEC 9797 specifies object identifiers that can be used to identify each mechanism in accordance with ISO/IEC 8825-1. Numerical examples and a security analysis of each of the six specified algorithms are provided, and the relationship of this part of ISO/IEC 9797 to previous standards is explained.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 18033-3, *Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
block
bit string of length n

3.2
block cipher key
key that controls the operation of a block cipher

3.3
ciphertext
data which has been transformed to hide its information content

[ISO/IEC 9798-1:2010]

3.4
data integrity
property that data has not been altered or destroyed in an unauthorized manner

[ISO 7498-2]

3.5
decryption
reversal of a corresponding encryption

[ISO/IEC 9798-1:2010]

3.6
encryption
reversible operation by a cryptographic algorithm converting data into ciphertext so as to hide the information content of the data

[ISO/IEC 9798-1:2010]

3.7
key
sequence of symbols that controls the operation of a cryptographic transformation

NOTE Examples are encryption, decryption, cryptographic check function computation, signature generation, or signature verification.

[ISO/IEC 9798-1:2010]

3.8
MAC algorithm key
key that controls the operation of a MAC algorithm

3.9
Message Authentication Code
MAC
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 [1]).

3.10
Message Authentication Code algorithm
MAC algorithm
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 a set of input strings and corresponding function values, where the value of the ith input string might have been chosen after observing the value of the first i -1 function values (for integers i > 1)

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

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

3.11
***n*-bit block cipher**
block cipher with the property that plaintext blocks and ciphertext blocks are n bits in length

[ISO/IEC 10116]

3.12
output transformation
function that is applied at the end of the MAC algorithm, before the truncation operation

3.13
plaintext
unencrypted information

NOTE Adapted from ISO/IEC 9798-1:2010.

4 Symbols and notation

Throughout this part of ISO/IEC 9797 the following symbols and notation are used:

CT_i n-bit binary representation of the integer i.

D data string to be input to the MAC algorithm.

D_j block derived from the data string D after the padding and splitting process.

$d_K(C)$ decryption of the ciphertext C with the block cipher e using the key K.

$e_K(P)$ encryption of the plaintext P with the block cipher e using the key K.

F final iteration.

g output transformation that maps the block H_q to the block G.

G block that is the result of the output transformation.

$\mathrm{GF}(2^n)$ finite field with exactly 2^n elements.

$H_0, H_1, \ldots, H_q$ blocks used in the MAC algorithm to store intermediate results.

k length (in bits) of the block cipher key.

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

K, K', K'' secret block cipher keys of length (in bits) k.

K_1, K_2 secret masking keys of length (in bits) n.

L length block, used in Padding Method 3, equal to the binary representation of the length of the input message, left-padded to form an n-bit block.

L_D length (in bits) of the data string D.

m length (in bits) of the MAC.

multx(T) operation on an n-bit string T defined as $T*x$, where T is treated as an element in the finite field $\mathrm{GF}(2^n)$, and is multiplied by the element corresponding to the monomial x in $\mathrm{GF}(2^n)$. It can be computed as follows, where T_{n-1} denotes the leftmost bit of T and, as defined below, $<<$ denotes a one-bit left shift operation.

$$\text{multx}(T) = \begin{cases} T \ll 1 & \text{if } T_{n-1} = 0 \\ (T \ll 1) \oplus \tilde{P}_n & \text{if } T_{n-1} = 1 \end{cases}$$

n block length (in bits) of the block cipher.

$p_n(x)$ irreducible polynomials of degree n over $\mathrm{GF}(2)$, that is, polynomials with no non-trivial divisors.

$\tilde{p}_n$ string of bits of length n, consisting of the rightmost n coefficients (corresponding to x^{n-1}, x^{n-2}, …, x, x^0 = 1) of the irreducible polynomial $p_n(x)$. For n=128, $p_n(x) = x^{128} + x^7 + x^2 + x + 1$, and $\tilde{p}_{128}$ = $0^{120}10000111$. For n=64, $p_n(x) = x^{64} + x^4 + x^3 + x + 1$, and $\tilde{p}_{64}$ = $0^{59}11011$.

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

S secret string of length (in bits) n.

S_1, S_2 secret strings of length (in bits) $t \cdot n$.

t smallest integer greater than or equal to k/n.

$j \sim X$ 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).

0^n string consisting of n zero bits.

:= 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.

$*$ finite field multiplication. In the polynomial representation, each element of $\mathrm{GF}(2^n)$ is represented by a binary polynomial of degree less than n. More explicitly the bit string $A = a_{n-1} \ldots a_2 a_1 a_0$ is mapped to the binary polynomial $a(x) = a_{n-1}x^{n-1} + \ldots + a_2x_2 + a_1x + a_0$. Multiplication in the finite field $\mathrm{GF}(2^n)$, denoted by $A*B$, corresponds to the multiplication of two polynomials $a(x)b(x)$ modulo a binary irreducible polynomial $p_n(x)$ of degree n; that is, $A*B$ is the polynomial of degree at most n-1 obtained by multiplying *a*(*x*) and *b*(*x*), dividing the result by $p_n(x)$, and then taking the remainder. Here $p_n(x)$ is chosen to be the lexicographically first polynomial from among the irreducible polynomials of degree n that have a minimum number of non-zero coefficients. For n=128, $p_n(x) = x^{128} + x^7 + x^2 + x + 1$.

X<<1 string obtained from the string X by a left shift of 1 bit; if the length of X is n bits then X<<1 is the string consisting of the rightmost n bits of $X \| 0$.

5 Requirements

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

— a block cipher e, either one of those specified in ISO/IEC 18033-3 or the DEA block cipher (specified in Annex A of ISO/IEC 18033-3:2005 and ANSI X3.92 [10]). DEA may only be used with MAC Algorithms 3 and 4;

— a padding method from amongst those specified in 6.3;

— a MAC algorithm from amongst those specified in Clause 7;

— the length (in bits) m of the MAC; and

— a common key derivation method if MAC Algorithm 4 is used; a common key derivation method may also be required for MAC Algorithms 2 and 6.

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

The length m of the MAC shall be a positive integer less than or equal to the block length n.

If Padding Method 3 is used, the length in bits of the data string D shall be less than 2^n.

If MAC Algorithm 4 is used, the number of blocks in the padded version of the data string shall be greater than or equal to two, i.e., $q \geq 2$.

The selection of a specific block cipher *e*, padding method, MAC algorithm, value for m, and key derivation method (if any) are beyond the scope of this part of ISO/IEC 9797.

NOTE 1 These choices affect the security level of the MAC algorithm. For a detailed discussion, see Annex C.

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

NOTE 2 It is considered to be good cryptographic practice to have independent keys for confidentiality and for data integrity.

The security of the MAC algorithms in this part of ISO/IEC 9797 is critically dependent on the procedures and practices followed to manage the keys. Information about key management can be found in ISO 8732 [3], ISO/IEC 11770 [8] and ISO 11568 [9].

Disclosure of intermediate values during the computation of the MAC algorithms may enable forgery and/or key recovery attacks (cf. Annex C).

6 Model for MAC algorithms

6.1 General

The application of the MAC algorithm requires the following seven steps: key derivation (optional), padding, splitting, iterative application of the block cipher, final iteration, output transformation, and truncation. Steps 4 through 7 are illustrated in Figure 1.

NOTE In MAC Algorithm 4, the first stage of the iteration (Step 4) is different from the other stages.

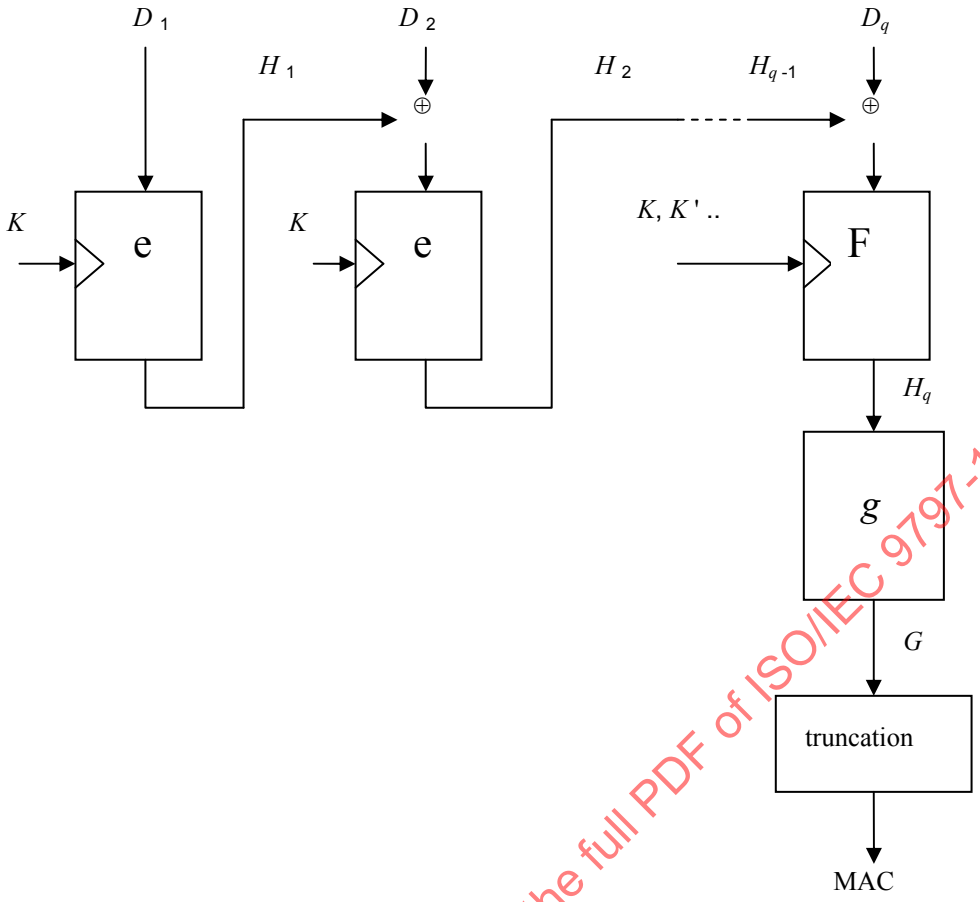

Figure 1 — Application of steps 4, 5, 6 and 7 of the MAC algorithm

6.2 Step 1 (key derivation)

6.2.1 General

MAC Algorithm 5 uses a key derivation algorithm which derives two masking keys from a block cipher key. MAC Algorithms 2, 4 and 6 may need a key derivation algorithm, which derives two block cipher keys from a block cipher key.

This part of ISO/IEC 9797 specifies two key derivation algorithms.

6.2.2 Key Derivation Method 1

This key derivation method computes two block cipher keys K' and K'', each of length (in bits) k, from a block cipher key K.

This key derivation method uses the Counter Method (CTR) defined in ISO/IEC 10116 [7]. It consists of the following operations:

— Define the integer t as the smallest integer greater than or equal to k/n.

— Define the counter CT_i, $1 \le i \le 2t$ as the string consisting of the binary representation of the integer i left-padded with as few (possibly none) '0' bits as necessary to obtain an n-bit block.

— Compute the string S_1 of length (in bits) tn equal to $e_K(CT_1)||e_K(CT_2)||\ldots||e_K(CT_t)$ and set $K' := k \sim S_1$.

— Compute the string S_2 of length (in bits) tn equal to $e_K(CT_{t+1})||e_K(CT_{t+2})||\ldots||e_K(CT_{2t})$ and set $K'' := k \sim S_2$.

6.2.3 Key Derivation Method 2

This key derivation method computes two masking keys K_1 and K_2 of length in bits n from a block cipher key.

It consists of the following operations:

— First the secret string S of length in bits n is computed as follows: $S := e_K(0^n)$.

— Next the masking key K_1 is obtained from S : $K_1 :=$ multx(S).

— Finally the masking key K_2 is derived from K_1 : $K_2 :=$ multx(K_1).

6.3 Step 2 (padding)

6.3.1 General

This step involves prefixing and/or postfixing the data string D with additional 'padding' bits such that the padded version of the data string will always be a multiple of n bits in length. The padding bits that are added to the original data string, according to the chosen padding method, 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, and which padding method is in use.

This part of ISO/IEC 9797 specifies four padding methods. Padding methods 1, 2 and 3 can be chosen for MAC Algorithms 1, 2, 3, 4, and 6 specified in this part of ISO/IEC 9797. Padding method 4 shall only be used with MAC Algorithm 5.

6.3.2 Padding Method 1

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 whose length (in bits) is a positive integer multiple of n.

NOTE 1 MAC algorithms using Padding Method 1 may be subject to trivial forgery attacks. See informative Annex C for further details.

NOTE 2 If the data string is empty, Padding Method 1 specifies that it is right-padded with n `0' bits.

6.3.3 Padding Method 2

The data string D to be input to the MAC algorithm shall be right-padded with a single '1' bit. The resulting string shall then be right-padded with as few (possibly none) '0' bits as necessary to obtain a data string whose length (in bits) is a positive integer multiple of n.

NOTE If the data string is empty, Padding Method 2 specifies that it is right-padded with a single '1' bit followed by n - 1 '0' bits.

6.3.4 Padding Method 3

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 whose length (in bits) is a positive integer multiple of n. The resulting string shall then be left-padded with a block L. The block L consists of the binary representation of the length (in bits) L_D of the unpadded data string D, left-padded with as few (possibly none) '0' bits as necessary to obtain an n-bit block. The right-most bit of the block L corresponds to the least significant bit of the binary representation of L_D.

NOTE 1 Padding Method 3 is not suitable for use in situations where the length of the data string is not available prior to the start of the MAC calculation.

NOTE 2 If the data string is empty, Padding Method 3 specifies that it is right-padded with n '0' bits and left-padded with a block L consisting of n '0' bits.

6.3.5 Padding Method 4

If the data string D to be input to the MAC algorithm has a length (in bits) that is a positive integer multiple of n, no padding shall be applied. Otherwise, the data string D shall be right-padded with a single '1' bit. The resulting string shall then be right-padded with as few (possibly none) '0' bits as necessary to obtain a data string whose length (in bits) is a positive integer multiple of n.

NOTE If the data string is empty, Padding Method 4 specifies that it is right-padded with a single '1' bit followed by n - 1 '0' bits.

6.4 Step 3 (splitting)

The padded version of the data string D is split into q n-bit blocks $D_1, D_2, \ldots, D_q$. Here D_1 represents the first n bits of the padded version of D, D_2 represents the next n bits, and so on.

6.5 Step 4 (iteration)

The blocks $H_1, H_2, \ldots, H_{q-1}$ are calculated by iteratively applying the block cipher with block cipher key K to the bitwise exclusive-or of the data block D_i and the previous result H_{i-1}:

H_0 := 0;

for i from 1 to q - 1:

$H_i := e_K(D_i \oplus H_{i-1})$;

If q is equal to 1, Step 4 shall be omitted.

NOTE This operation corresponds to CBC (Cipher Block Chaining) mode with starting variable fixed to 0^n, as defined in ISO/IEC 10116 [7].

6.6 Step 5 (final iteration)

6.6.1 General

The final iteration F is applied to the last block D_q of the padded data string to derive the block H_q.

Each of the six MAC algorithms specified in this part of ISO/IEC 9797 uses one of three possible final iterations.

6.6.2 Final iteration 1

This transformation uses the same block cipher key K as the iteration. The block H_q is computed by applying the block cipher with key K as follows:

$H_q := e_K(D_q \oplus H_{q-1})$.

6.6.3 Final iteration 2

This transformation uses a block cipher key K', which is different from the block cipher key K used in the iteration. The block H_q is computed by applying the block cipher with key K' as follows:

$H_q := e_{K'}(D_q \oplus H_{q-1})$.

6.6.4 Final iteration 3

This transformation uses the same block cipher key K as the iteration and two masking keys K_1 and K_2 of length n. The block H_q is computed by XORing to the input the key K_1 or K_2 depending on the padding operation, followed by an encryption with the block cipher key K.

Following Padding Method 4, if the data string to be input to the MAC algorithm had a length (in bits) that is a positive integer multiple of n then:

$$H_q := e_K (D_q \oplus H_{q-1} \oplus K_1),$$

else

$$H_q := e_K (D_q \oplus H_{q-1} \oplus K_2).$$

6.7 Step 6 (output transformation)

6.7.1 General

The output transformation g is applied to the value H_q, obtained as a result of Step 5.

This part of ISO/IEC 9797 specifies three output transformations.

6.7.2 Output Transformation 1

This output transformation is the identity function, i.e.,

$$G := H_q.$$

6.7.3 Output Transformation 2

This output transformation consists of applying the block cipher with block cipher key K' to H_q, i.e.,

$$G := e_{K'}(H_q).$$

6.7.4 Output Transformation 3

This output transformation consists of applying the block cipher (in decryption mode) with the key K' to H_q, followed by applying the block cipher with key K to the result of this operation, i.e.,

$$G := e_K(d_{K'}(H_q)).$$

6.8 Step 7 (truncation)

The MAC of m bits is derived by taking the leftmost m bits of the block G, i.e.,

$$\text{MAC} := m \sim G.$$

7 MAC algorithms

7.1 General

This part of ISO/IEC 9797 specifies six MAC algorithms. The final iteration and output transformation are specified in each case.

7.2 MAC Algorithm 1

MAC Algorithm 1 uses Final Iteration 1 and Output Transformation 1. The MAC algorithm key consists of the block cipher key K. MAC Algorithm 1 is illustrated in Figure 2.

MAC Algorithm 1 may be used with Padding Method 1, 2 or 3 specified in 6.3.

NOTE 1 The choice of padding method affects the security of the MAC algorithm. See informative Annex C for further details.

NOTE 2 MAC Algorithm 1 is subject to XOR forgery attacks (see Annex C). As a result this algorithm should only be used where such attacks are infeasible, e.g. because message lengths are fixed.

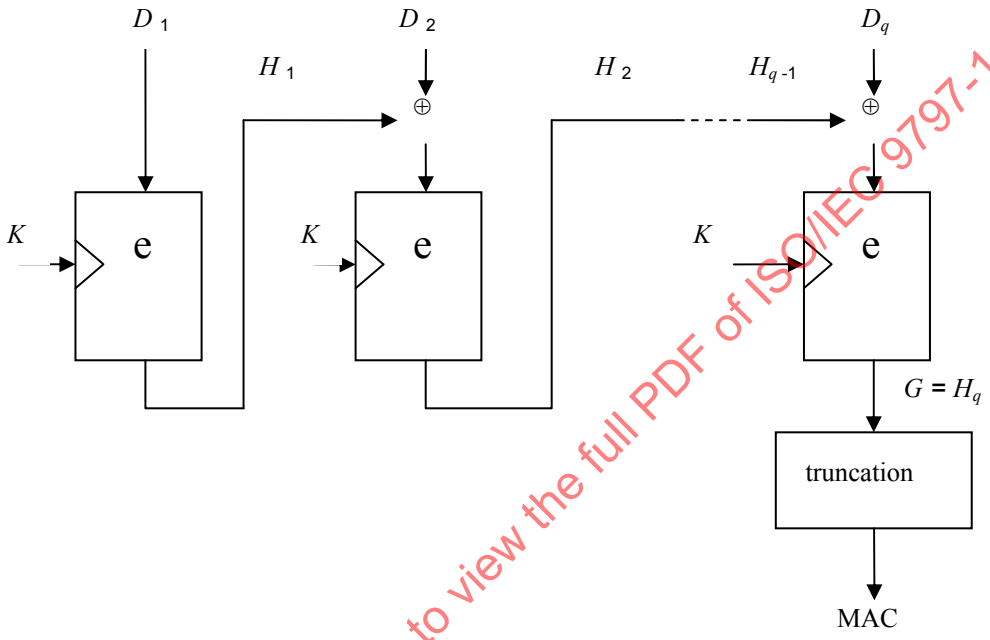

Figure 2 — MAC Algorithm 1

7.3 MAC Algorithm 2

MAC Algorithm 2 uses Final Iteration 1 and Output Transformation 2. The MAC algorithm key consists of two block cipher keys K and K'. The values of K and K' may be derived from a common master key (a block cipher key) in such a way that K and K' are different with very high probability.

NOTE 1 MAC Algorithm 2 is commonly known as EMAC [24].

NOTE 2 An example of how to derive K and K' from a common master key is Key Derivation Method 1.

NOTE 3 If K and K' are equal, a simple XOR forgery attack applies. See informative Annex C for further details.

NOTE 4 If K and K' are independent, the level of security against key recovery attacks is less than suggested by the MAC algorithm key size. See informative Annex C for further details.

MAC Algorithm 2 is illustrated in Figure 3.

MAC Algorithm 2 may be used with Padding Method 1, 2 or 3 specified in 6.3.

NOTE 5 The choice of padding method affects the security of the MAC algorithm. See informative Annex C for further details.

NOTE 6 If MAC Algorithm 2 is used in combination with an algorithm that computes a (public) key identifier as $S=e_K(0^n)$, such as X9.24 [13], then MAC Algorithm 2 is subject to XOR forgery attacks (see Annex C). In this case the algorithm should only be used where such attacks are infeasible, e.g. because message lengths are fixed.

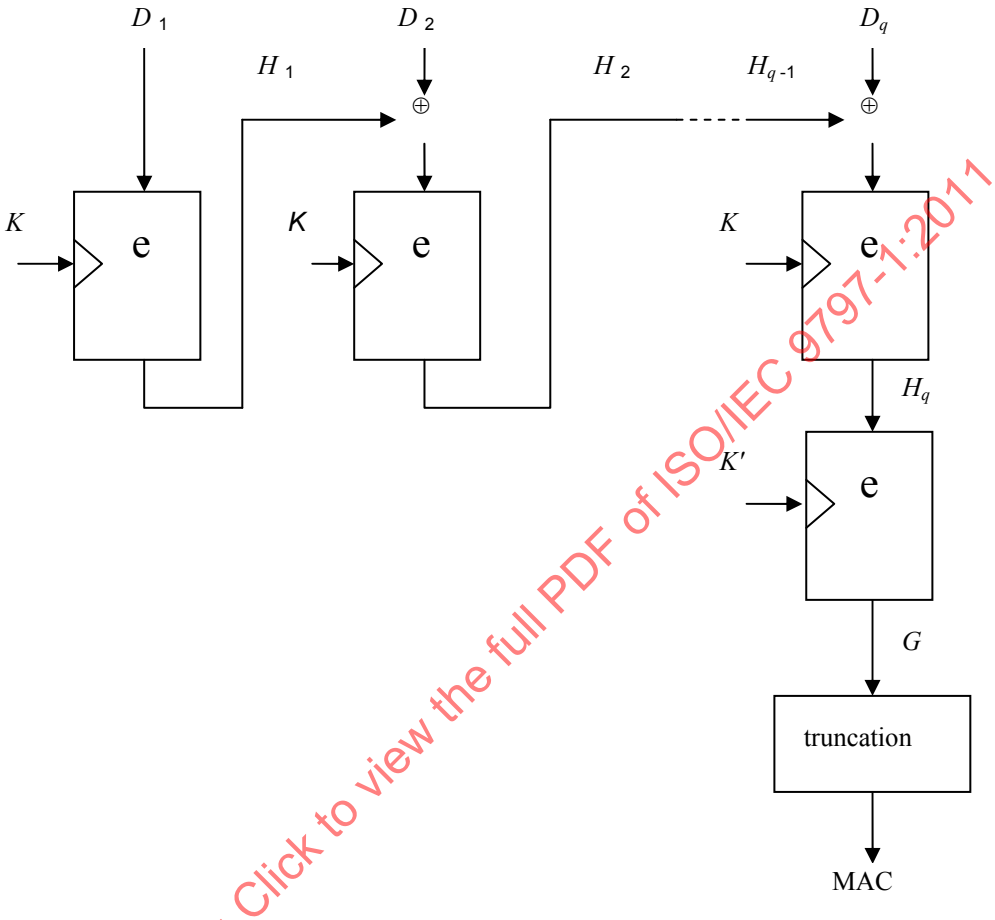

Figure 3 — MAC Algorithm 2

7.4 MAC Algorithm 3

MAC Algorithm 3 uses Final Iteration 1 and Output Transformation 3. The MAC algorithm key consists of two block cipher keys K and K'. The values of K and K' shall be chosen independently. If $K' = K$, MAC Algorithm 3 reduces to MAC Algorithm 1. MAC Algorithm 3 is illustrated in Figure 4.

NOTE 1 MAC Algorithm 3 is commonly known as the ANSI retail MAC [12]. When MAC Algorithm 3 is used with DEA (specified in Annex A of ISO/IEC 18033-3:2005 and ANSI X3.92 [10]), the block cipher key length is 56 bits, while the key length is equal to 112 bits.

MAC Algorithm 3 may be used with Padding Method 1, 2 or 3 specified in 6.3.

NOTE 2 The choice of padding method affects the security of the MAC algorithm. See informative Annex C for further details.

NOTE 3 If MAC Algorithm 3 is used in combination with an algorithm that computes a (public) key identifier as $S=e_K(0^n)$, such as X9.24 [13], then MAC Algorithm 3 is subject to XOR forgery attacks (see Annex C). In this case the algorithm should only be used where such attacks are infeasible, e.g. because message lengths are fixed.

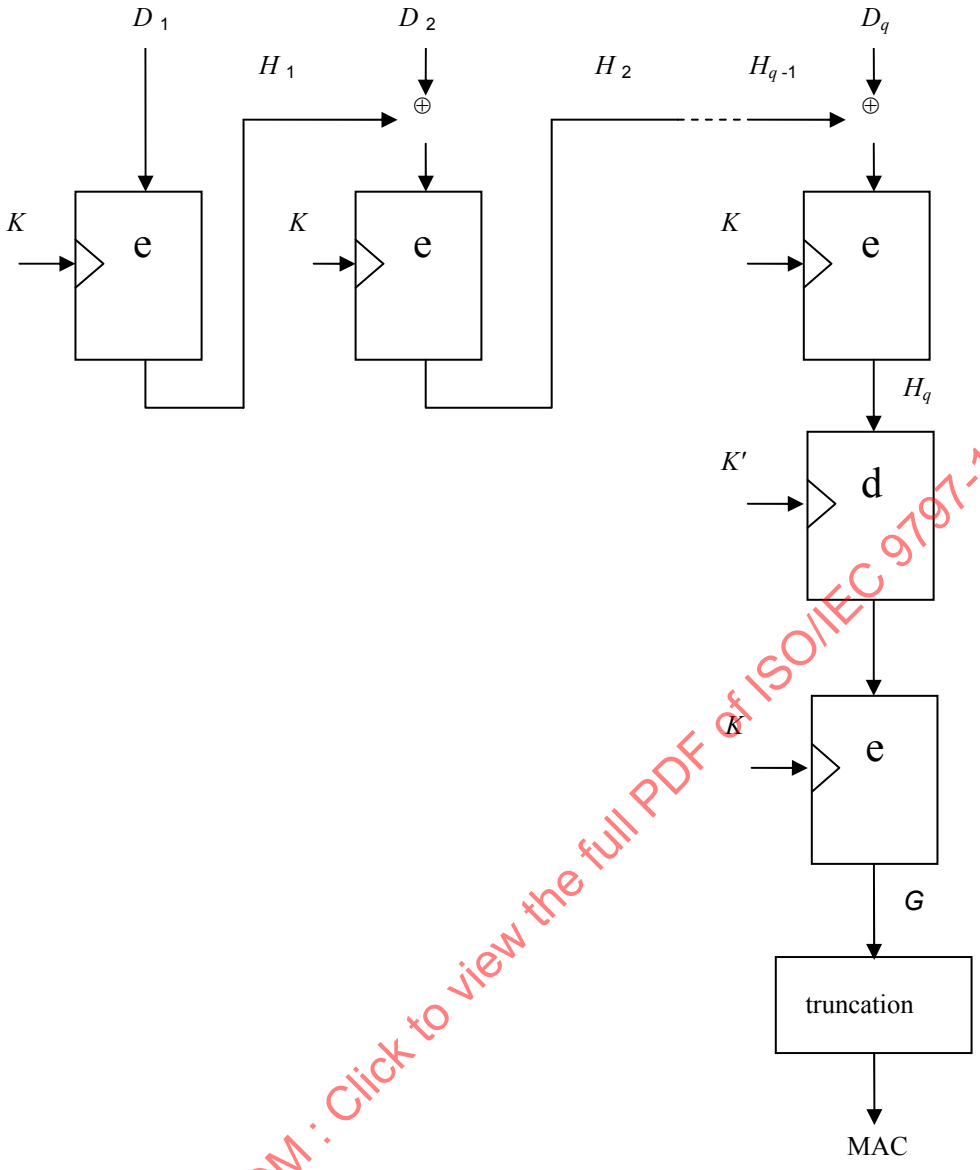

Figure 4 — MAC Algorithm 3

7.5 MAC Algorithm 4

MAC Algorithm 4 uses Final Iteration 1 and Output Transformation 2. In addition, MAC Algorithm 4 changes the processing of the first block.

NOTE 1 When used with DEA (specified in Annex A of ISO/IEC 18033-3:2005 and ANSI X3.92 [10]), MAC Algorithm 4 is known as MacDES [21]. In this case, the block cipher key length is 56 bits, while the key length is equal to 112 bits.

The MAC algorithm key consists of two block cipher keys K and K', that shall be chosen independently. The third block cipher key K'' shall be derived from K'. The values of K, K', and K'' shall be different. The block cipher keys K and K'' are used in the processing of the first block, and the block cipher keys K and K' are used with Output Transformation 2.

NOTE 2 An example of how to derive K'' from K' is to complement alternate substrings of four bits of K' commencing with the first four bits. Another example is to derive both K' and K'' from a common master key as specified in Key Derivation Method 1.

The first block is not processed using the normal iteration (with one encryption) but rather using the following equation (with two encryptions):

$$H_1 := e_{K''}(e_K(D_1)) .$$

MAC Algorithm 4 is illustrated in Figure 5.

MAC Algorithm 4 may be used with Padding Method 1, 2 or 3 specified in 6.3.

NOTE 3 The choice of padding method affects the security of the MAC algorithm. See informative Annex C for further details.

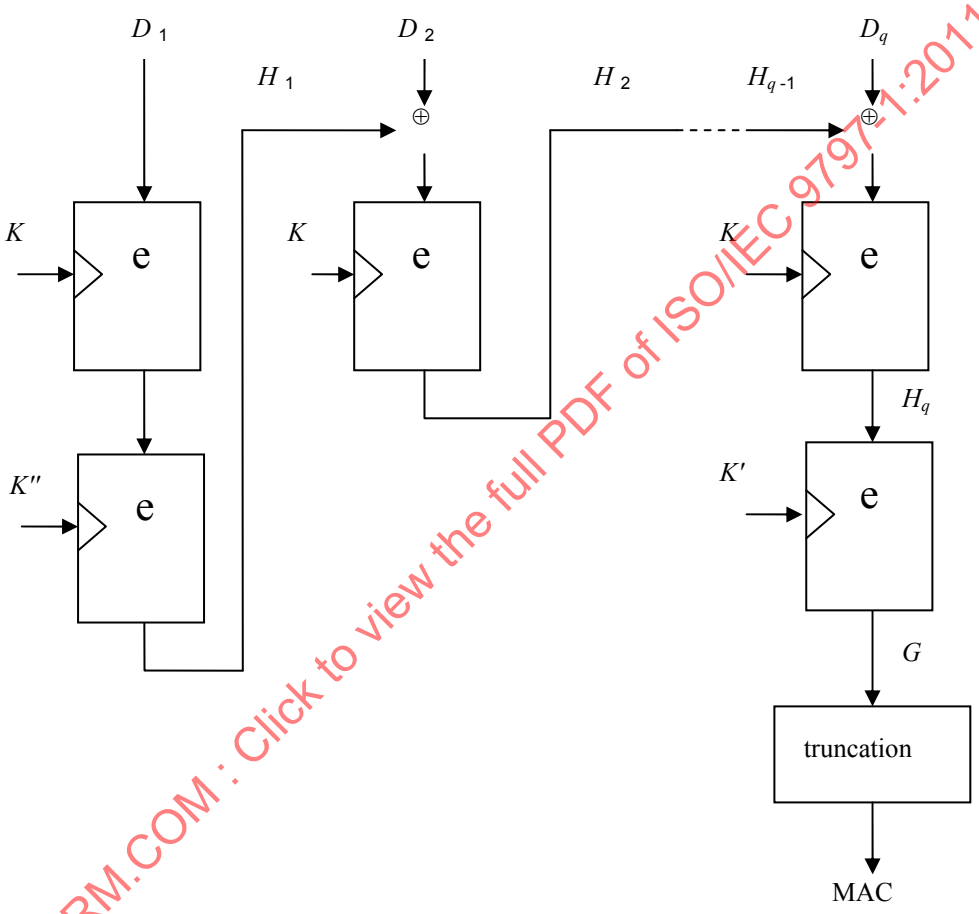

Figure 5 — MAC Algorithm 4

7.6 MAC Algorithm 5

MAC Algorithm 5 uses Key Derivation Method 2, Final Iteration 3 and Output Transformation 1. MAC Algorithm 5 shall only be used with Padding Method 4. The masking keys K_1 and K_2 to be used in Final Iteration 3 are derived from the MAC algorithm key K using Key Derivation Method 2.

NOTE 1 MAC Algorithm 5 is commonly known as OMAC1 [19] or CMAC [14].

The MAC algorithm key consists of a single block cipher key K.

MAC Algorithm 5 is illustrated in Figure 6, in which $K_i = K_1$ or K_2.

NOTE 2 If MAC Algorithm 5 is used in combination with an algorithm that computes a (public) key identifier as $S=e_K(0^n)$, such as X9.24 [13], then MAC Algorithm 5 is subject to XOR forgery attacks (see Annex C). In this case the algorithm should only be used where such attacks are infeasible, e.g. because message lengths are fixed.

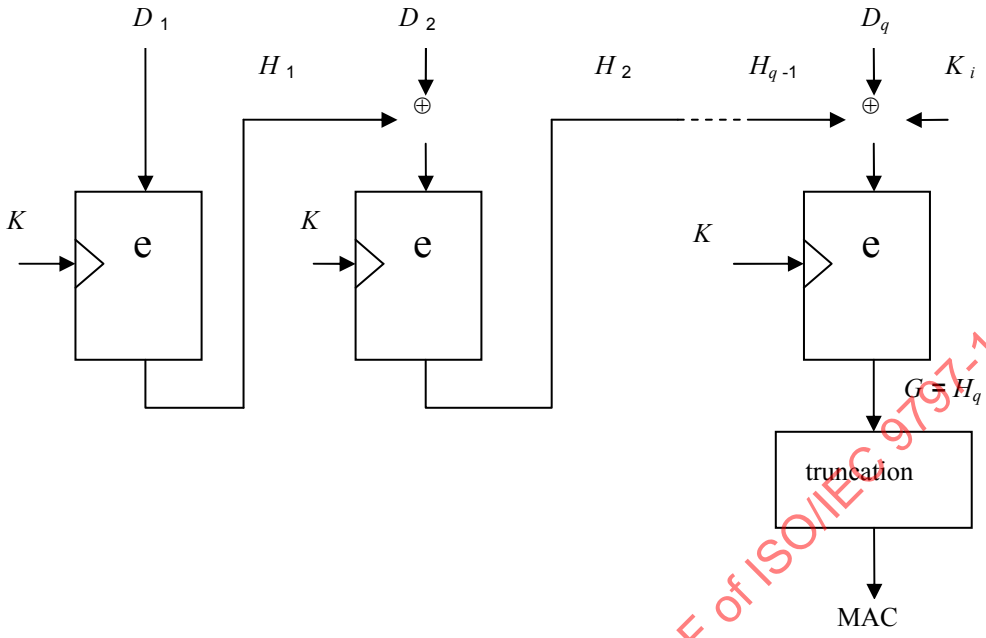

Figure 6 — MAC Algorithm 5

7.7 MAC Algorithm 6

MAC Algorithm 6 uses Final Iteration 2 and Output Transformation 1. The MAC algorithm key consists of two block cipher keys K and K'. The values of K and K' may be derived from a common master key (a block cipher key) in such a way that K and K' are different with very high probability.

NOTE 1 MAC Algorithm 6 is commonly known as LMAC.

NOTE 2 An example of how to derive K and K' from a common master key is Key Derivation Method 1.

NOTE 3 If K and K' are equal, a simple XOR forgery attack applies. See informative Annex C for further details.

NOTE 4 If K and K' are independent, the level of security against key recovery attacks is less than suggested by the MAC algorithm key size. See informative Annex C for further details.

MAC Algorithm 6 is illustrated in Figure 7.

MAC Algorithm 6 may be used with Padding Method 1, 2 or 3 specified in 6.3.

NOTE 5 The choice of padding method affects the security of the MAC algorithm. See informative Annex C for further details.

NOTE 6 If MAC Algorithm 6 is used in combination with an algorithm that computes a (public) key identifier as $S=e_K(0^n)$, such as X9.24 [13], then MAC Algorithm 6 is subject to XOR forgery attacks (see Annex C). In this case the algorithm should only be used where such attacks are infeasible, e.g. because message lengths are fixed.

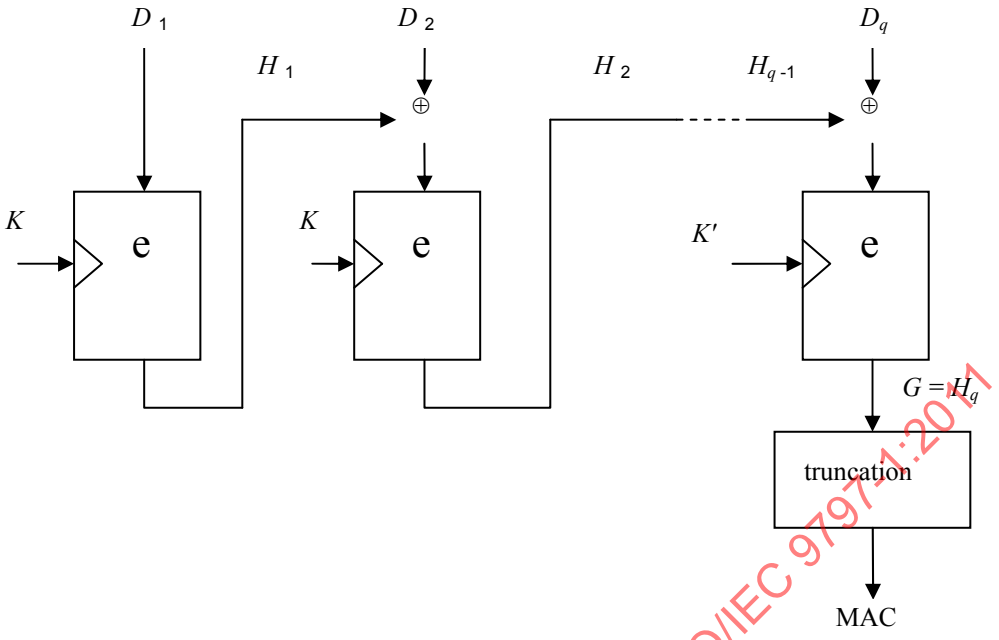

Figure 7 — MAC Algorithm 6

Annex A
(normative)

Object identifiers

```
MessageAuthenticationCodesPart1 {
    iso(1) standard(0) message-authentication-codes(9797) part(1)
        asn1-module(0) algorithm-object-identifiers(0)}

DEFINITIONS

EXPLICIT TAGS ::=

BEGIN

IMPORTS
    ALGORITHM, BlockAlgorithms
        FROM EncryptionAlgorithms-3 {iso(1) standard(0)
            encryption-algorithms(18033) part(3)
            asn1-module(0) algorithm-object-identifiers(0)};

OID ::= OBJECT IDENTIFIER
BASE-OID ::= OID

-- OID assignments
-- ===============
is9797-1 OID ::= {iso standard message-authentication-codes(9797) part1(1)}

id-kdm BASE-OID ::= {is9797-1 keyDerivationMethod(1)}
id-pm BASE-OID ::= {is9797-1 padMethod(2)}
id-ma BASE-OID ::= {is9797-1 macAlgo(3)}

-- normative comment:
-- concatenation of these relative OIDs and the id-kdm base OID specifies
-- a full object identifier for each of the specified key derivation methods
-- for possible use in other documents
id-kdm-1 RELATIVE-OID ::= {1}
id-kdm-2 RELATIVE-OID ::= {2}

-- normative comment:
-- concatenation of these relative OIDs and the id-pm base OID specifies
-- a full object identifier for each of the specified padding methods
-- for possible use in other documents
id-pad-1 RELATIVE-OID ::= {1}
id-pad-2 RELATIVE-OID ::= {2}
id-pad-3 RELATIVE-OID ::= {3}
id-pad-4 RELATIVE-OID ::= {4}

id-mac-1 OID ::= {id-ma 1}
id-mac-2 OID ::= {id-ma 2}
id-mac-3 OID ::= {id-ma 3}
id-mac-4 OID ::= {id-ma 4}
id-mac-5 OID ::= {id-ma 5}
id-mac-6 OID ::= {id-ma 6}
```

```
-- MAC algorithm identifier type and the set of recognized MAC algorithms
-- ========================================================================
MessageAuthenticationCode ::= AlgorithmIdentifier {{ MacAlgorithms }}

MacAlgorithms ALGORITHM ::= {
    { OID id-mac-1 PARMS MacParameters-1 } |
    { OID id-mac-2 PARMS MacParameters-2 } |
    { OID id-mac-3 PARMS MacParameters-3 } |
    { OID id-mac-4 PARMS MacParameters-4 } |
    { OID id-mac-5 PARMS MacParameters-5 } |
    { OID id-mac-6 PARMS MacParameters-6 } ,
    ... -- expect additional algorithms --
}

-- MAC parameter types definitions
-- ===============================

-- this makes it possible to specify an application defined key derivation method
KdAlgo ::= CHOICE {
    specifiedKdAlgo RELATIVE-OID,
    generalKdAlgo   OID
}

-- a parameter structure used in 5 of the 6 MAC algorithms
-- the optional parameters are either not used (eg. MAC algorithm 1 does not
-- use a key derivation method) or may be agreed upon by other means
MacParameters ::= SEQUENCE {
    bcAlgo  BlockCipher OPTIONAL,
    padAlgo [0] RELATIVE-OID ({1}|{2}|{3}) OPTIONAL,
    kdAlgo  [1] KdAlgo OPTIONAL,
    m       INTEGER (1..MAX)
}

MacParameters-1 ::= MacParameters

MacParameters-2 ::= MacParameters

MacParameters-3 ::= MacParameters

MacParameters-4 ::= MacParameters

-- for MAC algorithm 5 the padding method and the key derivation method are fixed
MacParameters-5 ::= SEQUENCE {
    bcAlgo  BlockCipher OPTIONAL,
    m       INTEGER (1..MAX)
}

MacParameters-6 ::= MacParameters


-- auxiliary definitions
-- =====================

-- definition of a block cipher algorithm identifier
BlockCipher ::= AlgorithmIdentifier {{BlockAlgorithms}}
```

```
AlgorithmIdentifier {ALGORITHM:IOSet} ::= SEQUENCE {
    algorithm ALGORITHM.&id({IOSet}),
    parameters ALGORITHM.&Type({IOSet}{@algorithm}) OPTIONAL
}

END -- MessageAuthenticationCodes --
```

Annex B
(informative)

Examples

B.1 General

This annex presents examples of the generation of a MAC.

For MAC Algorithms 1–4, the plaintexts are the 7-bit ASCII codes (no parity) for data string 1: "Now□is□the□time□for□all□" and data string 2: "Now□is□the□time□for□it", where "□" denotes a blank. ASCII coding is equivalent to coding using ISO 646. All MAC values and key values are written in hexadecimal notation.

For data string 1, the results of applying padding methods 1–3 are as follows:

— Padding Method 1: q = 3

D_1	4E 6F 77 20 69 73 20 74
D_2	68 65 20 74 69 6D 65 20
D_3	66 6F 72 20 61 6C 6C 20

— Padding Method 2: q = 4

D_1	4E 6F 77 20 69 73 20 74
D_2	68 65 20 74 69 6D 65 20
D_3	66 6F 72 20 61 6C 6C 20
D_4	80 00 00 00 00 00 00 00

— Padding Method 3: q = 4

D_1	00 00 00 00 00 00 00 C0
D_2	4E 6F 77 20 69 73 20 74
D_3	68 65 20 74 69 6D 65 20
D_4	66 6F 72 20 61 6C 6C 20

For data string 2, the results of applying padding methods 1–3 are as follows:

— Padding Method 1: q = 3

D_1	4E 6F 77 20 69 73 20 74
D_2	68 65 20 74 69 6D 65 20
D_3	66 6F 72 20 69 74 00 00

— Padding Method 2: q = 3

D_1	4E 6F 77 20 69 73 20 74
D_2	68 65 20 74 69 6D 65 20
D_3	66 6F 72 20 69 74 80 00

— Padding Method 3: q = 4

D_1	00 00 00 00 00 00 00 B0
D_2	4E 6F 77 20 69 73 20 74
D_3	68 65 20 74 69 6D 65 20
D_4	66 6F 72 20 69 74 00 00

B.2 MAC Algorithm 1

The examples given use DEA as the block cipher (specified in Annex A of ISO/IEC 18033-3:2005 and ANSI X3.92 [10]). The key value used is K = 0123456789ABCDEF (hexadecimal). The length m in bits of the MAC is equal to 32.

— Data string 1 with Padding Method 1

key (K)	01 23 45 67 89 AB CD EF
H_1	3F A4 0E 8A 98 4D 48 15
$D_2 \oplus H_1$	57 C1 2E FE F1 20 2D 35
H_2	0B 2E 73 F8 8D C5 85 6A
$D_3 \oplus H_2$	6D 41 01 D8 EC A9 E9 4A
$G = H_3$	70 A3 06 40 CC 76 DD 8B

MAC = 70 A3 06 40

— Data string 1 with Padding Method 2

key (K)	01 23 45 67 89 AB CD EF
H_1	3F A4 0E 8A 98 4D 48 15
$D_2 \oplus H_1$	57 C1 2E FE F1 20 2D 35
H_2	0B 2E 73 F8 8D C5 85 6A
$D_3 \oplus H_2$	6D 41 01 D8 EC A9 E9 4A
H_3	70 A3 06 40 CC 76 DD 8B
$D_4 \oplus H_3$	F0 A3 06 40 CC 76 DD 8B
$G = H_4$	10 E1 F0 F1 08 34 1B 6D

MAC = 10 E1 F0 F1

— Data string 1 with Padding Method 3

key (K)	01 23 45 67 89 AB CD EF
H_1	4B B5 82 65 DD 87 B3 05
$D_2 \oplus H_1$	05 DA F5 45 B4 F4 93 71
H_2	40 C4 00 AD 74 2E 4F D6
$D_3 \oplus H_2$	28 A1 20 D9 1D 43 2A F6
H_3	23 7D 5F 95 0B F7 1F 57
$D_4 \oplus H_3$	45 12 2D B5 6A 9B 73 77
$G = H_4$	2C 58 FB 8F F1 2A AE AC

MAC = 2C 58 FB 8F

— Data string 2 with Padding Method 1

key (K)	01 23 45 67 89 AB CD EF
H_1	3F A4 0E 8A 98 4D 48 15
$D_2 \oplus H_1$	57 C1 2E FE F1 20 2D 35
H_2	0B 2E 73 F8 8D C5 85 6A
$D_3 \oplus H_2$	6D 41 01 D8 E4 B1 85 6A
$G = H_3$	E4 5B 3A D2 B7 CC 08 56

MAC = E4 5B 3A D2

— Data string 2 with Padding Method 2

key (K)	01 23 45 67 89 AB CD EF
H_1	3F A4 0E 8A 98 4D 48 15
$D_2 \oplus H_1$	57 C1 2E FE F1 20 2D 35
H_2	0B 2E 73 F8 8D C5 85 6A
$D_3 \oplus H_2$	6D 41 01 D8 E4 B1 85 6A
$G = H_3$	A9 24 C7 21 36 14 92 11

MAC = A9 24 C7 21

— Data string 2 with Padding Method 3

key (K)	01 23 45 67 89 AB CD EF
H_1	DF 9C D6 EA 7E 5A E1 62
$D_2 \oplus H_1$	91 F3 A1 CA 17 29 C1 16
H_2	C7 6F B0 02 94 A4 19 BE
$D_3 \oplus H_2$	AF 0A 90 76 FD C9 7C 9E
H_3	83 02 28 FD 78 D7 BE 71
$D_4 \oplus H_3$	E5 6D 5A DD 11 A3 BE 71
$G = H_4$	B1 EC D6 FC 8B 37 C3 92

MAC = B1 EC D6 FC

B.3 MAC Algorithm 2

The examples given use DEA as the block cipher (specified in Annex A of ISO/IEC 18033-3:2005 and ANSI X3.92 [10]). The two key values used are K = 0123456789ABCDEF (hexadecimal), and K' is computed by complementing alternate substrings of four bits commencing with the first four bits. The length m in bits of the MAC is equal to 32.

The first q steps are identical to those of MAC Algorithm 1. The only difference is that Output Transformation 2 is applied instead of Output Transformation 1.

— Data string 1 with Padding Method 1

key (K')	F1 D3 B5 97 79 5B 3D 1F
G	10 F9 BC 67 A0 3C D5 D8

MAC = 10 F9 BC 67

— Data string 1 with Padding Method 2

key (K')	F1 D3 B5 97 79 5B 3D 1F
G	BE 7C 2A B7 D3 6B F5 B7

MAC = BE 7C 2A B7

— Data string 1 with Padding Method 3

key (K')	F1 D3 B5 97 79 5B 3D 1F
G	8E FC 8B C7 C2 72 6E 5C

MAC = 8E FC 8B C7

— Data string 2 with Padding Method 1

key (K')	F1 D3 B5 97 79 5B 3D 1F
G	21 5E 9C E6 D9 1B C7 FB

MAC = 21 5E 9C E6

— Data string 2 with Padding Method 2

key (K')	F1 D3 B5 97 79 5B 3D 1F
G	17 36 AC 1A 63 63 0E FB

MAC = 17 36 AC 1A

— Data string 2 with Padding Method 3

key (K')	F1 D3 B5 97 79 5B 3D 1F
G	05 38 26 96 27 4F B4 F0

MAC = 05 38 26 96

B.4 MAC Algorithm 3

The examples given use DEA as the block cipher (specified in Annex A of ISO/IEC 18033-3:2005 and ANSI X3.92 [10]). The two key values used are K = 0123456789ABCDEF (hexadecimal), and K' = FEDCBA9876543210 (hexadecimal). The length m in bits of the MAC is equal to 32.

The first q steps are identical to those of MAC Algorithm 1. The only difference is that Output Transformation 3 is applied instead of Output Transformation 1.

— Data string 1 with Padding Method 1

key (K')	FE DC BA 98 76 54 32 10
output of d	B4 8D 36 EC 7A D5 69 4F
G	A1 C7 2E 74 EA 3F A9 B6

MAC = A1 C7 2E 74

— Data string 1 with Padding Method 2

key (K')	FE DC BA 98 76 54 32 10
output of d	79 53 7F EE 18 CF 18 93
G	E9 08 62 30 CA 3B E7 96

MAC = E9 08 62 30

— Data string 1 with Padding Method 3

key (K')	FE DC BA 98 76 54 32 10
output of d	FE B3 B9 66 1D BE DE CD
G	AB 05 94 63 D7 A7 D1 70

MAC = AB 05 94 63

— Data string 2 with Padding Method 1

key (K')	FE DC BA 98 76 54 32 10
output of d	32 8A C7 8B A1 CA 0B 3F
G	2E 2B 14 28 CC 78 25 4F

MAC = 2E 2B 14 28

— Data string 2 with Padding Method 2

key (K')	FE DC BA 98 76 54 32 10
output of d	7A 71 AF 2F 5D 15 40 A7
G	5A 69 2C E6 4F 40 41 45

MAC = 5A 69 2C E6

— Data string 2 with Padding Method 3

key (K')	FE DC BA 98 76 54 32 10
output of d	20 97 B4 05 F1 9E 2D D8
G	C5 9F 7E ED 32 8D DD 69

MAC = C5 9F 7E ED

B.5 MAC Algorithm 4

The examples given use DEA as the block cipher (specified in Annex A of ISO/IEC 18033-3:2005 and ANSI X3.92 [10]). The two key values used are K = 0123456789ABCDEF (hexadecimal), and K' = FEDCBA9876543210 (hexadecimal). Derived keys are computed by complementing alternate substrings of four bits commencing with the first four bits. The length m in bits of the MAC is equal to 32.

— Data string 1 with Padding Method 1

key (K)	01 23 45 67 89 AB CD EF
key (K')	FE DC BA 98 76 54 32 10
key (K'')	0E 2C 4A 68 86 A4 C2 E0
output of e	3F A4 0E 8A 98 4D 48 15
H_1	EA F0 4B F5 31 ED 33 5E
$D_2 \oplus H_1$	82 95 6B 81 58 80 56 7E
H_2	7E 7F 98 A0 C8 B1 65 6C
$D_3 \oplus H_2$	18 10 EA 80 A9 DD 09 4C
H_3	7B 93 0A AE 67 4A C9 24
G	AD 35 02 B7 AC 4A 48 A0

MAC = AD 35 02 B7

— Data string 1 with Padding Method 2

key (K)	01 23 45 67 89 AB CD EF
key (K')	FE DC BA 98 76 54 32 10
key (K'')	0E 2C 4A 68 86 A4 C2 E0
output of e	3F A4 0E 8A 98 4D 48 15
H_1	EA F0 4B F5 31 ED 33 5E
$D_2 \oplus H_1$	82 95 6B 81 58 80 56 7E
H_2	7E 7F 98 A0 C8 B1 65 6C
$D_3 \oplus H_2$	18 10 EA 80 A9 DD 09 4C
H_3	7B 93 0A AE 67 4A C9 24
$D_3 \oplus H_3$	FB 93 0A AE 67 4A C9 24
H_4	26 C4 FA D7 2E 6D D3 A2

G	61 C3 33 E3 42 C5 53 7C

MAC = 61 C3 33 E3

— Data string 1 with Padding Method 3

key (K)	01 23 45 67 89 AB CD EF
key (K')	FE DC BA 98 76 54 32 10
key (K'')	0E 2C 4A 68 86 A4 C2 E0
output of e	4B B5 82 65 DD 87 B3 05
H_1	71 5A F8 BE DA BE 90 44
$D_2 \oplus H_1$	3F 35 8F 9E B3 CD B0 30
H_2	50 2A 04 42 6A 80 B6 0B
$D_3 \oplus H_2$	38 4F 24 36 03 ED D3 2B
H_3	AF 13 8C 54 99 9B 84 30
$D_3 \oplus H_3$	C9 7C FE 74 F8 F7 E8 10
H_4	7F 90 05 61 B4 2C CE D2
G	95 2A F8 38 98 9B 5C 00

MAC = 95 2A F8 38

— Data string~2 with Padding Method 1

key (K)	01 23 45 67 89 AB CD EF
key (K')	FE DC BA 98 76 54 32 10
key (K'')	0E 2C 4A 68 86 A4 C2 E0
output of e	3F A4 0E 8A 98 4D 48 15
H_1	EA E0 4B F5 31 ED 33 5E
$D_2 \oplus H_1$	82 95 6B 81 58 80 56 7E
H_2	7E 7F 98 A0 C8 B1 65 6C
$D_3 \oplus H_2$	18 10 EA 80 A1 C5 65 6C
H_3	21 FC 35 F2 B2 26 6C 9A
G	05 F1 08 4C 1D E3 A3 3D

MAC = 05 F1 08 4C

— Data string 2 with Padding Method 2

key (K)	01 23 45 67 89 AB CD EF
key (K')	FE DC BA 98 76 54 32 10

key (K'')	0E 2C 4A 68 86 A4 C2 E0
output of e	3F A4 0E 8A 98 4D 48 15
H_1	EA F0 4B F5 31 ED 33 5E
$D_2 \oplus H_1$	82 95 6B 81 58 80 56 7E
H_2	7E 7F 98 A0 C8 B1 65 6C
$D_3 \oplus H_2$	18 10 EA 80 A1 C5 65 6C
H_3	8F 76 9B 55 48 42 23 FD
G	A1 BC 09 31 52 BB 3E 0F

MAC = A1 BC 09 31

— Data string 2 with Padding Method 3

key (K)	01 23 45 67 89 AB CD EF
key (K')	FE DC BA 98 76 54 32 10
key (K'')	0E 2C 4A 68 86 A4 C2 E0
output of e	DF 9C D6 EA 7E 5A E1 62
H_1	82 61 94 52 C7 6D 04 F1
$D_2 \oplus H_1$	CC 0E E3 72 AE 1E 24 85
H_2	ED 33 1C 07 37 D6 B8 26
$D_3 \oplus H_2$	85 56 3C 73 5E BB DD 06
H_3	7C A1 DE 70 BB 1F 7F 07
$D_3 \oplus H_3$	1A CE AC 50 D2 6B 7F 07
H_4	40 B7 45 2E F3 CF 71 49
G	AF DE E0 F9 50 39 66 3D

MAC = AF DE E0 F9

B.6 MAC Algorithm 5

B.6.1 Examples of MAC generation process

Ten examples of the MAC generation process for this algorithm are provided. The underlying block cipher in these examples is either AES or triple DEA (TDEA) (both of which are specified in ISO/IEC 18033-3:2005).

Two examples are provided for each of the possible key lengths for these two ciphers, i.e. 128, 196 and 256 bits for AES, and two-key and three-key TDEA. Within each pair of examples, the MAC computation for two different messages is given, both using the same key. The generation of masking keys K_1 and K_2 from the key K is specified in each case, followed by the two examples of MAC generation.

All strings are represented in hexadecimal notation.

B.6.2 AES using a 128-bit key

The following 128-bit key and associated derived masking keys are used in both examples:

key (K)	2B 7E 15 16 28 AE D2 A6	AB F7 15 88 09 CF 4F 3C
$S = e_K(0^{128})$	7D F7 6B 0C 1A B8 99 B3	3E 42 F0 47 B9 1B 54 6F
K_1	FB EE D6 18 35 71 33 66	7C 85 E0 8F 72 36 A8 DE
K_2	F7 DD AC 30 6A E2 66 CC	F9 0B C1 1E E4 6D 51 3B

The MAC computations are as follows.

Data string (D)	The empty string	
G	BB 1D 69 29 E9 59 37 28	7F A3 7D 12 9B 75 67 46

Data string (D)	6B C1 BE E2 2E 40 9F 96	E9 3D 7E 11 73 93 17 2A
G	07 0A 16 B4 6B 4D 41 44	F7 9B DD 9D D0 4A 28 7C

B.6.3 AES using a 192-bit key

The following 192-bit key and associated derived masking keys are used in both examples:

key (K)	8E 73 B0 F7 DA 0E 64 52	C8 10 F3 2B 80 90 79 E5
	62 F8 EA D2 52 2C 6B 7B	
$S = e_K(0^{128})$	22 45 2D 8E 49 A8 A5 93	9F 73 21 CE EA 6D 51 4B
K_1	44 8A 5B 1C 93 51 4B 27	3E E6 43 9D D4 DA A2 96
K_2	89 14 B6 39 26 A2 96 4E	7D CC 87 3B A9 B5 45 2C

The MAC computations are as follows.

Data string (D)	The empty string	
G	D1 7D DF 46 AD AA CD E5	31 CA C4 83 DE 7A 93 67

Data string (D)	6B C1 BE E2 2E 40 9F 96	E9 3D 7E 11 73 93 17 2A
G	9E 99 A7 BF 31 E7 10 90	06 62 F6 5E 61 7C 51 84

B.6.4 AES using a 256-bit key

The following 256-bit key and associated derived masking keys are used in both examples:

key (K)	60 3D EB 10 15 CA 71 BE	2B 73 AE F0 85 7D 77 81
	1F 35 2C 07 3B 61 08 D7	2D 98 10 A3 09 14 DF F4
$S = e_K(0^{128})$	E5 68 F6 81 94 CF 76 D6	17 4D 4C C0 43 10 A8 54

K_1	CA D1 ED 03 29 9E ED AC	2E 9A 99 80 86 21 50 2F
K_2	95 A3 DA 06 53 3D DB 58	5D 35 33 01 0C 42 A0 D9

The MAC computations are as follows.

Data string (D)	The empty string	
G	02 89 62 F6 1B 7B F8 9E	FC 6B 55 1F 46 67 D9 83

Data string (D)	6B C1 BE E2 2E 40 9F 96	E9 3D 7E 11 73 93 17 2A
G	28 A7 02 3F 45 2E 8F 82	BD 4B F2 8D 8C 37 C3 5C

B.6.5 Three-key triple DEA

The following key and associated derived masking keys are used in both examples (where K is a triple of DEA keys):

key (K)	8A A8 3B F8 CB DA 10 62
	0B C1 BF 19 FB B6 CD 58
	BC 31 3D 4A 37 1C A8 B5
$S = e_K(0^{64})$	C8 CC 74 E9 8A 73 29 A2
K_1	91 98 E9 D3 14 E6 53 5F
K_2	23 31 D3 A6 29 CC A6 A5

The MAC computations are as follows.

Data string (D)	The empty string
G	B7 A6 88 E1 22 FF AF 95

Data string (D)	6B C1 BE E2 2E 40 9F 96
G	8E 8F 29 31 36 28 37 97

B.6.6 Two-key triple DEA

The following key and associated derived masking keys are used in both examples (where K is a triple of DEA keys, with the first and third keys the same):

key (K)	4C F1 51 34 A2 85 0D D5
	8A 3D 10 BA 80 57 0D 38
	4C F1 51 34 A2 85 0D D5
$S = e_K(0^{64})$	C7 67 9B 9F 6B 8D 7D 7A

K_1	8E CF 37 3E D7 1A FA EF
K_2	1D 9E 6E 7D AE 35 F5 C5

The MAC computations are as follows.

Data string (D)	The empty string
G	BD 2E BF 9A 3B A0 03 61

Data string (D)	6B C1 BE E2 2E 40 9F 96
G	4F F2 AB 81 3C 53 CE 83

B.7 MAC Algorithm 6

B.7.1 Examples of MAC generation process

Three examples of the MAC generation process for this algorithm are provided. The underlying block cipher in these examples is AES (specified in ISO/IEC 18033-3:2005). Padding Method 2 is used.

One example is provided for each possible key length for this cipher, i.e. 128, 196 and 256 bits. The generation of K and K' from a single key K^* is specified in each case, followed by the example of MAC generation.

All strings are represented in hexadecimal notation.

B.7.2 AES using a 128-bit key

The following keys K and K' are used in the example (where K and K' are derived from K^* using key derivation method 1):

K^*	91 18 69 5B E6 B7 86 F2	81 7A BE FB 54 E2 58 29
K	0D D9 B7 C6 0C 9F 1E E0	63 D6 BB 3E 4F E5 6B D9
K'	B7 9F 0C 87 04 1F 68 18	B6 CE 3F 3B 77 EE BE 08

The MAC computation is as follows.

D_1	61 62 63 80 00 00 00 00	00 00 00 00 00 00 00 00
G	E7 A8 FD 3F 6A 4F DB 80	33 1E E2 6E 94 09 CB 22

B.7.3 AES using a 192-bit key

The following keys K and K' are used in the example (where K and K' are derived from K^* using key derivation method 1):

K^*	C6 D0 9C CE 02 F8 34 70	E0 CF AE 90 17 90 A0 92
	41 8A AC B1 28 72 FE 9D	
K	1A D9 8F 06 2C 00 46 81	01 97 1B C0 19 8C E5 F0
	58 42 E3 73 D4 D4 82 A5	

K'	95 31 D7 D1 2B 8F 3E 8C	F8 B6 A9 CE E9 97 6B 11
	37 83 9F 7C 5D C6 6A A3	

The MAC computation is as follows.

D_1	48 65 6C 6C 6F 20 57 6F	72 6C 64 80 00 00 00 00
G	A5 C5 AD EC D5 4B DA 85	4E A8 DD FF FD A5 05 1F

B.7.4 AES using a 256-bit key

The following keys K and K' are used in the example (where K and K' are derived from K^* using key derivation method 1):

K^*	78 3D 99 0F 8A DA 0F E2	E2 EC 43 19 B4 90 F8 9D
	B2 9A D0 7A 41 ED 6D 75	E3 50 76 F2 C6 85 2E E1
K	64 76 71 37 61 40 3E FC	10 EC 83 5B EC 67 C3 EB
	FF 10 F3 82 BC 19 9A EB	8E E4 B6 66 71 6C C4 DC
K'	5B 59 59 9E D8 27 F9 2F	AC 0C F3 D4 69 AE 64 5B
	C6 40 1D 3C 32 0C 1D E9	2C 4C E2 F9 02 D3 E6 36

The MAC computation is as follows.

D_1	53 69 78 74 65 65 6E 20	4C 65 74 74 65 72 73 2E
D_2	80 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
G	A8 3E 5B 7E D6 C8 FD 25	62 F2 7C C1 FA 3F 55 A2

Annex C
(informative)

A security analysis of the MAC algorithms

C.1 General

This Annex discusses the security level of the MAC algorithms in this part of ISO/IEC 9797. Its goal is to assist the user of this part of ISO/IEC 9797 in selecting one of the mechanisms.

It will be assumed that the key length of the block cipher is k bits, while the key length of the MAC algorithm is equal to k^* bits. The value of k^* is thus equal to k or to $2k$.

In this Annex, $\text{MAC}_K(D)$ denotes the MAC for a string D computed using the MAC algorithm key K.

In order to determine the security level of a MAC algorithm, two attack strategies are considered:

— **forgery attack**: This attack consists of predicting the value of $\text{MAC}_K(D)$ for a data string D without initial knowledge of K. If the adversary can do this for a single data string, he is said to be capable of a *forgery*. Practical attacks often require that a forgery is *verifiable*, i.e., that the forged MAC is known to be correct beforehand with probability near 1. Moreover, in many applications the data string has a specific format, which imposes additional constraints on the data string D.

— **key recovery attack**: This attack consists of finding the MAC algorithm key K itself from a number of data string/MAC pairs. Such an attack is more powerful than forgery, since it allows for arbitrary forgeries. Note that a key length of 56 bits (as of the DEA algorithm) is no longer considered to offer sufficient protection for most applications.

The feasibility of an attack depends on the number of known and chosen data string/MAC pairs required, and on the number of off-line encryptions.

Possible attacks against MAC algorithms are described below; there is no guarantee that this list is exhaustive. The first two attacks are generic, i.e., they apply to any MAC algorithm. The next attack applies to any iterated MAC algorithm. The following three attacks are specific to one or more of the MAC algorithms described in this part of ISO/IEC 9797 (for more details see [15], [20], [21], [25], [26], [27]).

— **guessing the MAC**: This is a forgery which is not verifiable, and which has a success probability of $\max(1/2^m, 1/2^{k^*})$. This attack applies to all MAC algorithms, and can only be precluded by a judicious choice of m and k^*.

— **brute force key recovery**: This attack should require on average 2^{k^*-1} operations; verification of such an attack requires of the order of k^*/m data string/MAC pairs. Again this attack applies to all MAC algorithms. It can be precluded by a judicious choice of the value k^*. Alternatively, one can prevent someone obtaining the k^*/m data string/MAC pairs which are necessary to identify the key uniquely. For example, if $k^* = 64$ and $m = 32$, approximately 2^{32} keys correspond to a given data string/MAC pair; if the key is changed after every data string, a brute force key recovery is no more effective than guessing the MAC value.

— **birthday forgery** [25], [27]**:** If one collects approximately $2^{n/2}$ data string/MAC pairs, this set will contain with high probability two data strings D and D' such that $\text{MAC}_K(D) = \text{MAC}_K(D')$ and the values of H_q in both computations are equal; this is called an internal collision. If D and D' form an internal collision, $\text{MAC}_K(D||Y) = \text{MAC}_K(D' || Y)$ for any string Y. This allows for a forgery after one chosen data string, as an adversary can predict the MAC for $D' || Y$ after having observed the MAC corresponding to $D||Y$. This

forgery is again on data strings of a specific form, which may not be a concern in all applications, but it should be noted that extensions of this attack exist which allow for greater flexibility in the data strings. The attack requires one chosen data string, and approximately $2^{n/2}$ known data strings and min{2^{n-m}, $2^{n/2}$} chosen data strings.

Note that the birthday forgery attack cannot be precluded by the combination of Padding Method 3 and the prepending of a block to the data string which contains a serial number (see [16] for more details).

— **trivial forgery**: If Padding Method 1 is used, an adversary can typically add or delete a number of trailing '0' bits of the data string without changing the MAC. This implies that Padding Method 1 shall only be used in environments where the length of the data string D is known to the parties beforehand, or where data strings with a different number of trailing '0' bits have the same semantics.

— **XOR forgery**: If MAC Algorithm 1 is used with Padding Method 1 or 2 and $m = n$, a simple XOR forgery is possible. Assume, for simplicity, that D has the property that its padded version $\overline{D}$ consists of a single block (hence, if Padding Method 2 is used, we are assuming that D is of length less than n bits). In addition let v denote the mapping that removes from a bit string the rightmost one, and all the zeros that follow this bit (and hence if $\overline{v(X)}$ denotes the padded version of $v(X)$ using Padding Method 2, then $\overline{v(X)} = X$).

Assume that one knows $\text{MAC}_K(D)$. If Padding Method 1 is used then it follows immediately that $\text{MAC}_K(\overline{D} \,||(\overline{D} \oplus \text{MAC}_K(D))) = \text{MAC}_K(D)$. Similarly, if Padding Method 2 is used, then it follows that $\text{MAC}_K(\overline{D} \,||v(\overline{D} \oplus \text{MAC}_K(D))) = \text{MAC}_K(D)$. This implies that one can construct a new message with the same MAC value, which is a forgery.

Note that this attack applies even if a MAC algorithm key is used only once. Assume that one knows $\text{MAC}_K(D)$ and $\text{MAC}_K(D')$. If Padding Method 1 is used, then a similar calculation shows that $\text{MAC}(\overline{D} \,||\, \overline{D'} \oplus \text{MAC}_K(D))) = \text{MAC}_K(D')$ (here D can be of arbitrary length but D' must be one block long). Similarly, if Padding Method 2 is used, then it follows that $\text{MAC}(\overline{D} \,||v(\overline{D'} \oplus \text{MAC}_K(D))) = \text{MAC}_K(D')$ (here D can be of arbitrary length but $\overline{D'}$ must be one block long).

Additionally, for Padding Method 1, if one knows $\text{MAC}_K(D)$, $\text{MAC}_K(D||Y)$, and $\text{MAC}_K(D')$, one knows that $\text{MAC}_K(D' \,||\, Y') = \text{MAC}_K(D||Y)$ if $Y' = Y \oplus \text{MAC}_K(D) \oplus \text{MAC}_K(D')$ (if D and Y fall on block boundaries). This also allows for a forgery, as an adversary can forge the MAC on $D' \,||\, Y'$ given knowledge of the MACs for two known data strings and one chosen data string. A similar (but slightly more complex) forgery attack also works for Padding Method 2.

Note that all the above forgeries are on data strings of a specific form, which may not be a concern in all applications.

This attack can be precluded by using Padding Method 3.

This attack can be extended to the case $m<n$, but it becomes more difficult: in that case it requires knowledge of the MACs for an additional $2^{(n-m)/2}$ chosen data strings [20].

The same attack applies when MAC Algorithm 2 is used with two equal keys, i.e., $K'=K$. In this case the attack works when Y contains at least two blocks, and the first n bits of Y are '0' bits.

— **shortcut key recovery**: Some MAC algorithms are potentially vulnerable to key recovery attacks based on an internal collision. Examples are MAC Algorithm 3 (see [21], [22], [26]) and MAC Algorithm 4 in combination with Padding Method 1 or 2 [18] or Padding Method 3 [17]. MAC Algorithm 5 allows for a partial key recovery attack [23]; once part of the key has been obtained, finding forgeries becomes easy.

The following tables present a comparison of the security level of the MAC algorithms described in this part of ISO/IEC 9797. It is assumed that the block cipher has no weaknesses. Table C.1 indicates the main properties of the MAC algorithms. As Padding Method 1 allows for a trivial forgery, the comparison involving MAC Algorithms 1, 2, 3, 4 and 6 considers only Padding Methods 2 and 3. Tables C.2 and C.3 present the