INTERNATIONAL STANDARD

ISO/IEC 9797-1

First edition
1999-12-15

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 (MACs) —

Partie 1: Mécanismes utilisant un cryptogramme bloc

Reference number
ISO/IEC 9797-1:1999(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Foreword

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

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

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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.

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

This first edition of ISO/IEC 9797-1, together with the subsequent parts of ISO/IEC 9797, cancels and replaces ISO/IEC 9797:1994, which has been revised and extended to a multi-part standard. Note, however, that implementations which comply with ISO/IEC 9797:1994 will be compliant with this edition of ISO/IEC 9797-1.

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 hash-function*

Further parts may follow.

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

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. 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 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 three mechanisms specified in this part of ISO/IEC 9797 are commonly known as CBC-MAC (CBC is the abbreviation of Cipher Block Chaining). The calculation of a MAC as described in ISO 8731-1 and ANSI X9.9 is a specific case of this part of ISO/IEC 9797 when $n = 64$, $m = 32$, MAC Algorithm 1 and Padding Method 1 are used, and the block cipher is DEA (ANSI X3.92: 1981). The calculation of a MAC as described in ANSI X9.19 and ISO 9807 is a specific case of this part of ISO/IEC 9797 when $n = 64$, $m = 32$, either MAC Algorithm 1 or MAC Algorithm 3 is used (both with Padding Method 1) and the block cipher is DEA (ANSI X3.92: 1981).

The fourth mechanism is a variant of CBC-MAC with a special initial transformation. It is recommended for applications which require that the key length of the MAC algorithm is twice that of the block cipher.

NOTES

1 For example, in the case of DEA (ANSI X3.92: 1981), the block cipher key length is 56 bits, while the MAC algorithm key length is 112 bits.

2 When used with DEA (which is also known as DES), this algorithm is called MacDES [12].

The fifth and sixth mechanism use two parallel instances of the first and fourth mechanism respectively, and combine the two results with a bitwise exclusive-or operation. They are recommended for applications which require an increased security level against forgery attacks (cf. Annex B). The fifth mechanism uses a single length MAC algorithm key, while the sixth mechanism doubles the MAC algorithm key length.

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 standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9797. At the time of publication, the editions indicated were valid. All standards are subject to revision, and 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 standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

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

ISO/IEC 9798-1: 1997, *Information technology — Security techniques — Entity authentication — Part 1: General.*

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

3 Definitions

3.1 This part of ISO/IEC 9797 makes use of the following general security-related term defined in ISO 7498-2.

3.1.1 data integrity: the property that data has not been altered or destroyed in an unauthorized manner.

3.2 For the purposes of this part of ISO/IEC 9797, the following definitions apply.

3.2.1 block: a bit-string of length n.

3.2.2 block cipher key: a key that controls the operation of a block cipher.

3.2.3 initial transformation: a function that is applied at the beginning of the MAC algorithm.

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

3.2.5 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).

3.2.6 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.

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

3.3 This part of ISO/IEC 9797 makes use of the following general security-related terms defined in ISO/IEC 9798-1.

3.3.1 ciphertext: data which has been transformed to hide its information content.

3.3.2 decipherment: the reversal of a corresponding encipherment.

3.3.3 encipherment: the (reversible) transformation of data by a cryptographic algorithm to produce ciphertext, i.e., to hide the information content of the data.

3.3.4 key: a sequence of symbols that controls the operation of a cryptographic transformation (e.g., encipherment, decipherment, cryptographic check function computation, signature generation, or signature verification).

3.3.5 plaintext: unenciphered information.

3.4 This part of ISO/IEC 9797 makes use of the following general security-related term defined in ISO/IEC 10116.

3.4.1 n-bit block cipher: a block cipher with the property that plaintext blocks and ciphertext blocks are n bits in length.

4 Symbols and notation

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

D data string to be input to the MAC algorithm.

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

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

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

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

G the block that is the result of the output transformation.

H_j a block which is used in the MAC algorithm to store an intermediate result.

I initial transformation.

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

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

K, K', K'', K''', K_1, K_2, K'_1, K'_2, K''_1, K''_2 secret block cipher keys.

L the length block, which is used in Padding Method 3.

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

m the length (in bits) of the MAC.

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

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.

5 Requirements

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

- a block cipher e;
- a padding method from amongst those specified in Clause 6.1;
- 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 algorithms 4, 5, and 6 are used; a common key derivation method may also be required for MAC algorithm 2.

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.

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

The same key shall be used for calculating and verifying the MAC. 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 Model for MAC algorithms

The application of the MAC algorithm requires the following six steps: padding, splitting, initial transformation, iterative application of the block cipher, output transformation, and truncation. Steps 3 through 6 are illustrated in Figure 1.

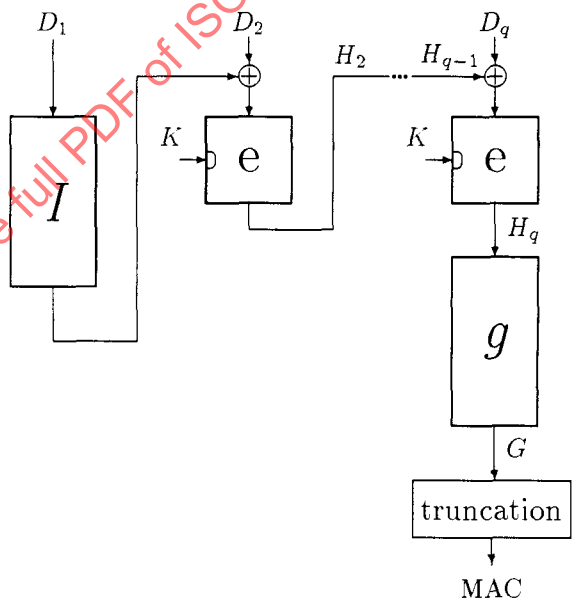

Figure 1: Application of Step 3, 4, 5 and 6 of the MAC algorithm.

6.1 Step 1 (padding)

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 three padding methods. Any of these three methods can be chosen for the six MAC algorithms specified in this part of ISO/IEC 9797.

6.1.1 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.

NOTES

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

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

6.1.2 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.

6.1.3 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 — 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.

6.2 Step 2 (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.

6.3 Step 3 (initial transformation)

The initial transformation I is applied to the first block D_1 of the padded data string to derive the block H_1.

Each of the six MAC algorithms specified in this part of ISO/IEC 9797 use one of two possible initial transformations.

6.3.1 Initial Transformation 1

This transformation requires only one block cipher key K. The block H_1 is computed by applying the block cipher with key K as follows:

$$H_1 := e_K(D_1)\,.$$

6.3.2 Initial Transformation 2

This transformation requires two block cipher keys K and K''. The block H_1 is computed by applying the block cipher with keys K and K'' as follows:

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

6.4 Step 4 (iteration)

The blocks $H_2, H_3, \ldots, H_q$ are calculated by iteratively applying the block cipher to the bitwise exclusive-or of the data block D_i and the previous result H_{i-1}:

for i from 2 to q:

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

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

NOTE — This operation corresponds to the Cipher Block Chaining (CBC) mode as defined in ISO/IEC 10116.

6.5 Step 5 (output transformation)

The output transformation g is applied to the value H_q, obtained as a result of Step 4 (or Step 3 in the case $q = 1$).

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

6.5.1 Output Transformation 1

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

$$G := H_q\,.$$

6.5.2 Output Transformation 2

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

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

6.5.3 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.,

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

6.6 Step 6 (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

This part of ISO/IEC 9797 specifies six MAC algorithms. The initial transformation and output transformation are specified in each case. However, the padding method is not specified, i.e., each MAC algorithm may be used with any of the three padding methods specified in Clause 6.1.

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

7.1 MAC Algorithm 1

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

7.2 MAC Algorithm 2

MAC Algorithm 2 uses Initial Transformation 1 and Output Transformation 2. The MAC algorithm key consists of two block cipher keys K and K'''. The value of K''' may be derived from the value of K in such a way that K and K''' are different.

NOTES

1 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.

2 If K and K''' are equal, a simple xor forgery attack applies. See informative Annex B for further details.

3 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 B for further details.

MAC Algorithm 2 is illustrated in Figure 3.

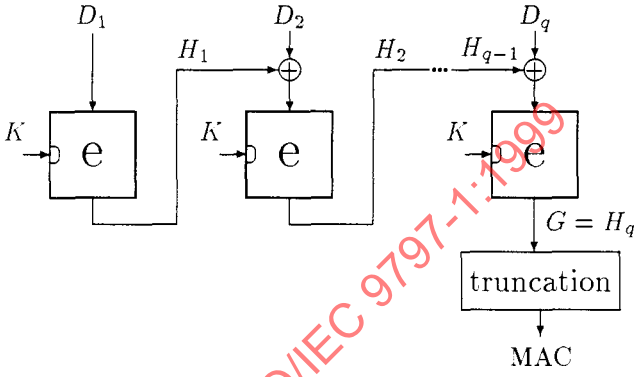

Figure 2 — MAC Algorithm 1.

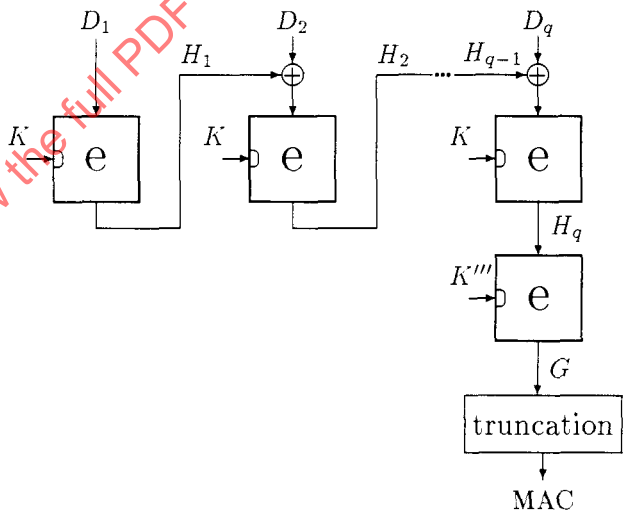

Figure 3 — MAC Algorithm 2.

7.3 MAC Algorithm 3

MAC Algorithm 3 uses Initial Transformation 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, which may not always be desirable. MAC Algorithm 3 is illustrated in Figure 4.

7.4 MAC Algorithm 4

MAC Algorithm 4 uses Initial Transformation 2 and Output Transformation 2. The MAC algorithm key con-

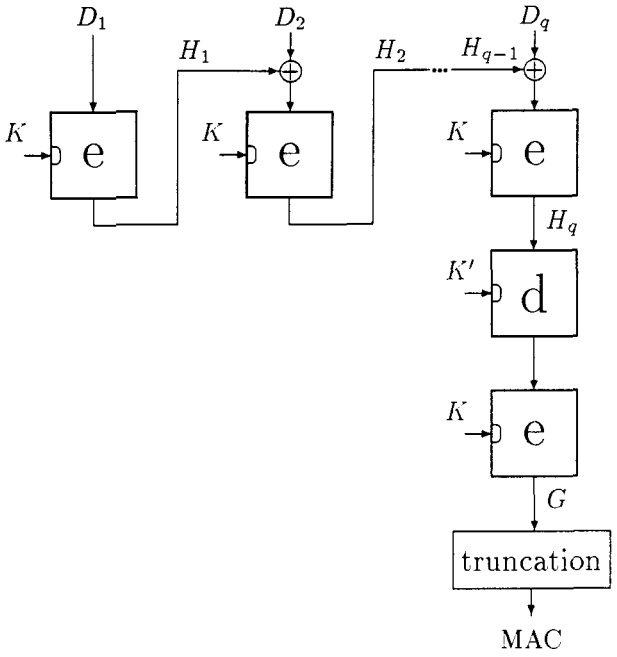

Figure 4 — MAC Algorithm 3.

sists of two block cipher keys K and K', that shall be chosen independently. A 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 with Initial Transformation 2, and the block cipher keys K and K' are used with Output Transformation 2.

NOTE — 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.

The number of blocks in the padded version of the data string shall be greater than or equal to two, i.e., $q \geq 2$.

MAC Algorithm 4 is illustrated in Figure 5.

7.5 MAC Algorithm 5

MAC Algorithm 5 uses two parallel instances of MAC Algorithm 1, resulting in two intermediate values, MAC_1 and MAC_2 respectively. The MAC algorithm key consists of the block cipher key K. The keys K_1 and K_2 used for the first and second instances are derived from the key K. The values of K_1 and K_2 shall be different.

NOTE — An example of how to derive K_1 and K_2 from K is to take K_1 equal to K and to construct K_2 by complementing alternate substrings of four bits of K commencing with the first four bits. Another example is to derive both K_1 and K_2 from a common master key in such a way that they are different.

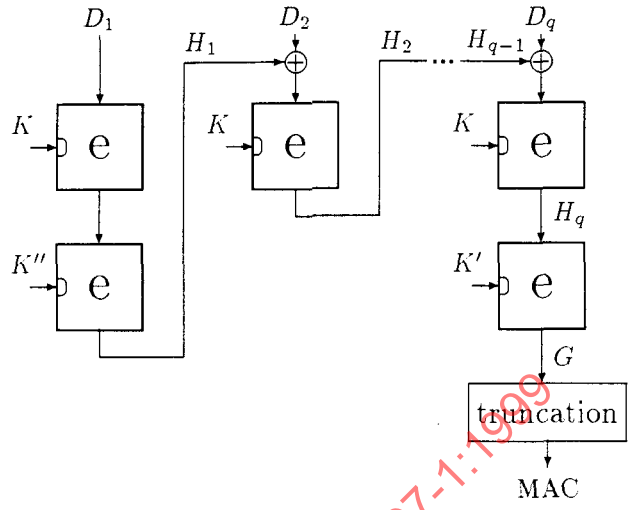

Figure 5 — MAC Algorithm 4.

The final MAC is obtained by a bitwise exclusive-or between intermediate values MAC_1 and MAC_2, i.e.,

$$\mathrm{MAC} := \mathrm{MAC}_1 \oplus \mathrm{MAC}_2 \ .$$

7.6 MAC Algorithm 6

MAC Algorithm 6 uses two parallel instances of MAC Algorithm 4, resulting in two intermediate values, MAC_1 and MAC_2 respectively. The MAC algorithm key consists of two block cipher keys K and K', that shall be chosen independently.

The keys (K_1, K_1') and (K_2, K_2') used for the first and second instances respectively shall be derived from the keys (K, K') in such a way that K_1 and K_1' are different, K_2 and K_2' are different, and the pairs (K_1, K_1') and (K_2, K_2') are different.

NOTES

1 An example of how to derive (K_1, K_1') and (K_2, K_2') from (K, K') is to take K_1 equal to K, K_1' equal to K', and to construct K_2 (K_2') by complementing alternate substrings of eight bits of K_1 (K_1') commencing with the first eight bits.

2 MAC Algorithm 4 internally uses a derived key K'', which means that MAC Algorithm 6 uses in total six block cipher keys (K_1, K_1', K_1'') and (K_2, K_2', K_2''). It is recommended to check that all of these keys are different.

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 final MAC is obtained by a bitwise exclusive-or between intermediate values MAC_1 and MAC_2, i.e.,

$$\mathrm{MAC} := \mathrm{MAC}_1 \oplus \mathrm{MAC}_2 \ .$$

Annex A
(informative)

Examples

This annex presents examples of the generation of a MAC using the DEA (see ANSI X3.92). 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. The two key values used are K = `0123456789ABCDEF` (hexadecimal), and K' = `FEDCBA9876543210` (hexadecimal). The key parity bits are ignored. Derived keys are computed by complementing alternate substrings of four bits commencing with the first four bits. For MAC Algorithms 1, 2, 3, and 4, $m = 32$, while for MAC Algorithms 5 and 6, $m = 64$. All values are written in hexadecimal notation.

For data string 1, the results of the three padding methods 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 the three padding methods 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

A.1 MAC Algorithm 1

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	05	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`

A.2 MAC Algorithm 2

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`

A.3 MAC Algorithm 3

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

A.4 MAC Algorithm 4

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	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	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	E5	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`

A.5 MAC Algorithm 5

The first part of the calculation is identical to that of Clause A.1.

Data string 1 with Padding Method 1

key (K_1)	01	23	45	67	89	AB	CD	EF
MAC_1	70	A3	06	40	CC	76	DD	8B
key (K_2)	FE	DC	BA	98	76	54	32	10
MAC_2	84	47	04	F6	7B	5A	CE	9C
$MAC_1 \oplus MAC_2$	F4	E4	02	B6	B7	2C	13	17

MAC = `F4 E4 02 B6 B7 2C 13 17`

Data string 1 with Padding Method 2

key (K_1)	01	23	45	67	89	AB	CD	EF
MAC_1	10	E1	F0	F1	08	34	1B	6D
key (K_2)	FE	DC	BA	98	76	54	32	10
MAC_2	60	11	AE	38	EC	C3	34	F4
$MAC_1 \oplus MAC_2$	70	F0	5E	C9	E4	F7	2F	99

MAC = `70 F0 5E C9 E4 F7 2F 99`

Data string 1 with Padding Method 3

key (K_1)	01	23	45	67	89	AB	CD	EF
MAC_1	2C	58	FB	8F	F1	2A	AE	AC
key (K_2)	FE	DC	BA	98	76	54	32	10
MAC_2	FA	47	AA	7D	1B	00	83	CF
$MAC_1 \oplus MAC_2$	D6	1F	51	F2	EA	2A	2D	63

MAC = `D6 1F 51 F2 EA 2A 2D 63`

Data string 2 with Padding Method 1

key (K_1)	01	23	45	67	89	AB	CD	EF
MAC_1	E4	5B	3A	D2	B7	CC	08	56
key (K_2)	FE	DC	BA	98	76	54	32	10
MAC_2	EB	7F	87	76	1B	EE	07	19
$MAC_1 \oplus MAC_2$	0F	24	BD	A4	AC	22	0F	4F

MAC = `0F 24 BD A4 AC 22 0F 4F`

Data string 2 with Padding Method 2

key (K_1)	01	23	45	67	89	AB	CD	EF
MAC_1	A9	24	C7	21	36	14	92	11
key (K_2)	FE	DC	BA	98	76	54	32	10
MAC_2	49	20	D4	60	AC	E8	84	1A
$MAC_1 \oplus MAC_2$	E0	04	13	41	9A	FC	16	0B

MAC = `E0 04 13 41 9A FC 16 0B`

Data string 2 with Padding Method 3

key (K_1)	01	23	45	67	89	AB	CD	EF
MAC_1	B1	EC	D6	FC	8B	37	C3	92
key (K_2)	FE	DC	BA	98	76	54	32	10
MAC_2	6C	33	88	2F	84	2F	28	6E
$MAC_1 \oplus MAC_2$	DD	DF	5E	D3	0F	18	EB	FC

MAC = `DD DF 5E D3 0F 18 EB FC`

A.6 MAC Algorithm 6

The first part of the calculation is identical to that of Clause A.4.

Data string 1 with Padding Method 1

key (K_1)	01	23	45	67	89	AB	CD	EF
key (K_1')	FE	DC	BA	98	76	54	32	10
key (K_1'')	0E	2C	4A	68	86	A4	C2	E0
MAC_1	AD	35	02	B7	AC	4A	48	A0
key (K_2)	FE	23	BA	67	76	AB	32	EF
key (K_2')	01	DC	45	98	89	54	CD	10
key (K_2'')	F1	2C	B5	68	79	A4	3D	E0
MAC_2	FA	4B	F0	96	B4	84	15	1A
$MAC_1 \oplus MAC_2$	57	7E	F2	21	18	CE	5D	BA

MAC = 57 7E F2 21 18 CE 5D BA

Data string 1 with Padding Method 2

key (K_1)	01	23	45	67	89	AB	CD	EF
key (K_1')	FE	DC	BA	98	76	54	32	10
key (K_1'')	0E	2C	4A	68	86	A4	C2	E0
MAC_1	61	C3	33	E3	42	C5	53	7C
key (K_2)	FE	23	BA	67	76	AB	32	EF
key (K_2')	01	DC	45	98	89	54	CD	10
key (K_2'')	F1	2C	B5	68	79	A4	3D	E0
MAC_2	01	B7	53	5B	9A	05	AE	86
$MAC_1 \oplus MAC_2$	60	74	60	B8	D8	C0	FD	FA

MAC = 60 74 60 B8 D8 C0 FD FA

Data string 1 with Padding Method 3

key (K_1)	01	23	45	67	89	AB	CD	EF
key (K_1')	FE	DC	BA	98	76	54	32	10
key (K_1'')	0E	2C	4A	68	86	A4	C2	E0
MAC_1	95	2A	F8	38	98	9B	5C	00
key (K_2)	FE	23	BA	67	76	AB	32	EF
key (K_2')	01	DC	45	98	89	54	CD	10
key (K_2'')	F1	2C	B5	68	79	A4	3D	E0
MAC_2	68	17	43	56	69	FE	5B	54
$MAC_1 \oplus MAC_2$	FD	3D	BB	6E	F1	65	07	54

MAC = FD 3D BB 6E F1 65 07 54

Data string 2 with Padding Method 1

key (K_1)	01	23	45	67	89	AB	CD	EF
key (K_1')	FE	DC	BA	98	76	54	32	10
key (K_1'')	0E	2C	4A	68	86	A4	C2	E0
MAC_1	05	F1	08	4C	1D	E3	A3	3D
key (K_2)	FE	23	BA	67	76	AB	32	EF
key (K_2')	01	DC	45	98	89	54	CD	10
key (K_2'')	F1	2C	B5	68	79	A4	3D	E0
MAC_2	15	06	4F	9D	52	91	61	14
$MAC_1 \oplus MAC_2$	10	F7	47	D1	4F	72	C2	29

MAC = 10 F7 47 D1 4F 72 C2 29

Data string 2 with Padding Method 2

key (K_1)	01	23	45	67	89	AB	CD	EF
key (K_1')	FE	DC	BA	98	76	54	32	10
key (K_1'')	0E	2C	4A	68	86	A4	C2	E0
MAC_1	A1	BC	09	31	52	BB	3E	0F
key (K_2)	FE	23	BA	67	76	AB	32	EF
key (K_2')	01	DC	45	98	89	54	CD	10
key (K_2'')	F1	2C	B5	68	79	A4	3D	E0
MAC_2	13	27	93	47	8F	A7	07	1D
$MAC_1 \oplus MAC_2$	B2	9B	9A	76	DD	1C	39	12

MAC = B2 9B 9A 76 DD 1C 39 12

Data string 2 with Padding Method 3

key (K_1)	01	23	45	67	89	AB	CD	EF
key (K_1')	FE	DC	BA	98	76	54	32	10
key (K_1'')	0E	2C	4A	68	86	A4	C2	E0
MAC_1	AF	DE	E0	F9	50	39	66	3D
key (K_2)	FE	23	BA	67	76	AB	32	EF
key (K_2')	01	DC	45	98	89	54	CD	10
key (K_2'')	F1	2C	B5	68	79	A4	3D	E0
MAC_2	59	9B	1B	84	1D	73	24	89
$MAC_1 \oplus MAC_2$	F6	45	FB	7D	4D	4A	42	B4

MAC = F6 45 FB 7D 4D 4A 42 B4

Annex B
(informative)

A security analysis of the MAC algorithms

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 $k^\star$ bits. The value of $k^\star$ is thus equal to k or to $2k$.

In this Annex, $\mathrm{MAC}_K(D)$ 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 $\mathrm{MAC}_K(D)$ 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.

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 encipherments.

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 for one or more of the MAC algorithms described in this part of ISO/IEC 9797 (for more details see [8, 11, 12, 14, 15, 16]).

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

brute force key recovery: this attack should require on average $2^{k^\star-1}$ operations; verification of such an attack requires of the order of $k^\star/m$ data string/MAC pairs. Again this attack applies to all MAC algorithms. It can be precluded by a judicious choice of the value $k^\star$. Alternatively, one can prevent someone obtaining the $k^\star/m$ data string/MAC pairs which are necessary to identify the key uniquely. For example, if $k^\star = 64$ 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 [14, 16]: if an adversary knows a sufficient number of data string/MAC pairs, he expects to find two data strings D and D' such that $\mathrm{MAC}_K(D) = \mathrm{MAC}_K(D')$ *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, $\mathrm{MAC}_K(D\|Y) = \mathrm{MAC}_K(D'\|Y)$ 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 2^{n-m} chosen data strings.

The birthday forgery attack can be precluded by the combination of the following measures: use Padding Method 3, prepend a block to the data string which contains a serial number and make the MAC computation stateful. This means that the implementation has to guarantee that each serial number is used only once for a MAC calculation during the lifetime of the key. This is not feasible in all environments. If one of the two measures is used and not the other, then variants of the basic birthday forgery attack still apply and have similar complexities; for example, if Padding Method 3 is used without the other measures, then one variant of the birthday attack (based on Lemma 1 of [16]) requires $2^{n/2}$ chosen plaintexts.

trivial forgery: if Padding Method 1 is used, an adver-

sary 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 or its padded version $\overline{D}$ consist of a single block. Assume that one knows $\mathrm{MAC}_K(D)$; it follows immediately that $\mathrm{MAC}_K(\overline{D}||(D \oplus \mathrm{MAC}_K(D))) = \mathrm{MAC}_K(D)$. 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. If one knows $\mathrm{MAC}_K(D)$ and $\mathrm{MAC}_K(D')$, a similar calculation shows that $\mathrm{MAC}(\overline{D}||(D' \oplus \mathrm{MAC}_K(D))) = \mathrm{MAC}_K(D')$. If one knows $\mathrm{MAC}_K(D)$, $\mathrm{MAC}_K(D||Y)$, and $\mathrm{MAC}_K(D')$, one knows that $\mathrm{MAC}_K(D'||Y') = \mathrm{MAC}_K(D||Y)$ if $Y' = Y \oplus \mathrm{MAC}_K(D) \oplus \mathrm{MAC}_K(D')$ (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. Note that 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 [11].

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 [12, 15]); and MAC Algorithm 4 in combination with Padding Method 1 or 2 [9] or Padding Method 3 [10].

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 1 indicates the main properties of the MAC algorithms. As Padding Method 1 allows for a trivial forgery, the comparison considers only Padding Methods 2 and 3. Table 2 and Table 3 present the best known attacks for block ciphers with $n = 64$ and $k = 56$ (e.g., the Data Encryption Algorithm [5]). Most of these attacks are described in [8, 9, 10, 11, 12, 14, 15, 16]; the shortcut key recovery attacks on MAC Algorithm 3 $m = 32$ (number 3 in Table 3) are obtained by an unpublished (but rather simple) extension of the corresponding attacks in Table 2. Table 4 and 5 are for block ciphers with $n = 128$ and $k = 128$. In this case only MAC Algorithms 1, 2, and 5 are considered, as there is no need to double the key length of the MAC algorithm. An attack is described as a four-tuple $[\alpha, \beta, \gamma, \delta]$, where α denotes the number of off-line block cipher encipherments, β denotes the number of known data string/MAC pairs, γ denotes the number of chosen data string/MAC pairs, and δ denotes the number of on-line MAC verifications.

In [8, 13], a lower bound is proved on the security level of CBC-MAC based on certain properties of the block cipher. This suggests that many of the birthday attacks discussed are close to the best attacks possible provided that the block cipher is strong. Reference [13] provides a security argument for MAC Algorithm 2.

Rationale

This section explains the selection of the MAC algorithms in this part of ISO/IEC 9797. An important factor in the selection of the proposed mechanism has been to maintain backward compatibility with the previous ANSI, ISO, and ISO/IEC standards.

Compared with the old version, the number of padding schemes has been increased from two to three. The third padding scheme provides resistance against certain attacks at very low cost, especially in applications where it is not a problem to prepend the length to the information.

The optional processes of the previous normative Annex A have been integrated into the main body.

Three new schemes have been included:

- MAC Algorithm 4 provides an improved way of increasing the key length. It is *strongly recommended* to use this algorithm in combination with Padding Method 3; in that case it provides better security than MAC Algorithm 3 at a comparable cost;
- MAC Algorithm 5 provides increased resistance against birthday forgery attacks;
- MAC Algorithm 6 provides both an increased key length and an increased resistance against birthday forgery attacks.

While several alternative ways exist to improve the basic CBC-MAC, the proposed solutions not only provide additional security, but are relatively simple, which makes them easy to analyse and implement.

Table 1: Properties of MAC algorithms. # Keys denotes the number of independent block cipher keys. The efficiency denotes the number of encipherments to process a data string of tn bits.

number	MAC Algorithm	Initial Transf.	Output Transf.	Padding	# Keys	efficiency
1.1	1	1	1	2	1	$t+1$
1.2	1	1	1	3	1	$t+2$
2.1	2	1	2	2	1	$t+2$
2.2	2	1	2	2	2	$t+2$
3	3	1	3	2	2	$t+3$
4.1	4	2	2	2	2	$t+3$
4.2	4	2	2	3	2	$t+4$
5	5	1	1	2	1	$2t+2$
6	6	2	2	2	2	$2t+6$

Table 2: Estimated security levels for $n = 64$, $k = 56$, and $m = 64$; the security level is determined by four numbers: off-line block cipher encipherments, known data string/MAC pairs, chosen data string/MAC pairs, and on-line MAC verifications

	key recovery		forgery		
number	brute force	shortcut	guess	xor	birthday
1.1	$[2^{56}, 1, 0, 0]$	—	$[0, 0, 0, 2^{56}]$	$[0, 1, 0, 0]$	$[0, 2^{32}, 1, 0]^{\dagger}$
1.2	$[2^{56}, 1, 0, 0]$	—	$[0, 0, 0, 2^{56}]$	—	$[0, 2^{32}, 1, 0]^{\dagger}$
2.1	$[2^{56}, 1, 0, 0]$	—	$[0, 0, 0, 2^{56}]$	—	$[0, 2^{32}, 1, 0]^{\dagger}$
2.2	$[2^{112}, 1, 0, 0]$	$[2^{57}, 2, 0, 0]$	$[0, 0, 0, 2^{64}]$	—	$[0, 2^{32}, 1, 0]^{\dagger}$
3	$[2^{112}, 2, 0, 0]$	$[2^{57}, 2^{32}, 0, 0]$ $[2^{56}, 1, 0, 2^{56}]$	$[0, 0, 0, 2^{64}]$ $[0, 1, 0, 2^{56}]$	—	$[0, 2^{32}, 1, 0]^{\dagger}$
4.1	$[2^{112}, 2, 0, 0]$	$[2^{58}, 2^{32}, 2, 0]^{\dagger}$ $[2^{58}, 1, 1, 2^{56}]^{\dagger}$	$[0, 0, 0, 2^{64}]$ $[2^{58}, 1, 1, 2^{56}]^{\dagger}$	—	$[0, 2^{32}, 1, 0]^{\dagger}$
4.2	$[2^{112}, 2, 0, 0]$	$[2^{57}, 2^{32}, 2^{50}, 0]$	$[0, 0, 0, 2^{64}]$	—	$[0, 2^{32}, 1, 0]^{\dagger}$
5	$[2^{56}, 1, 0, 0]$	—	$[0, 0, 0, 2^{56}]$	—	—
6	$[2^{112}, 2, 0, 0]$	—	$[0, 0, 0, 2^{64}]$	—	$[0, 2^{64}, 2^{64}, 0]^{\dagger}$

† Can be precluded by prepending a serial number to the data string in combination with Padding Method 3

Table 3: Estimated security levels for $n = 64$, $k = 56$, and $m = 32$; the security level is determined by four numbers: off-line block cipher encipherments, known data string/MAC pairs, chosen data string/MAC pairs, and on-line MAC verifications

	key recovery		forgery		
number	brute force	shortcut	guess	xor	birthday
1.1	$[2^{56}, 2, 0, 0]$	—	$[0, 0, 0, 2^{32}]$	$[0, 2, 2^{16}, 0]$	$[0, 2^{32}, 2^{32}, 0]^{\dagger}$
1.2	$[2^{56}, 2, 0, 0]$	—	$[0, 0, 0, 2^{32}]$	—	$[0, 2^{32}, 2^{32}, 0]^{\dagger}$
2.1	$[2^{56}, 2, 0, 0]$	—	$[0, 0, 0, 2^{32}]$	—	$[0, 2^{32}, 2^{32}, 0]^{\dagger}$
2.2	$[2^{112}, 4, 0, 0]$	$[2^{57}, 2^{32}, 2^{32}, 0]$ $[2^{88}, 4, 0, 0]$	$[0, 0, 0, 2^{32}]$	—	$[0, 2^{32}, 2^{32}, 0]^{\dagger}$
3	$[2^{112}, 4, 0, 0]$	$[2^{57}, 2^{32}, 2^{32}, 0]^{\dagger}$ $[2^{89}, 2^{32}, 0, 0]$ $[2^{56}, 2, 0, 2^{56}]$	$[0, 0, 0, 2^{32}]$	—	$[0, 2^{32}, 2^{32}, 0]^{\dagger}$
4.1	$[2^{112}, 4, 0, 0]$	$[2^{78}, 2^{32}, 2^{50}, 0]$	$[0, 0, 0, 2^{32}]$	—	$[0, 2^{32}, 2^{32}, 0]^{\dagger}$
4.2	$[2^{112}, 4, 0, 0]$	$[2^{78}, 2^{32}, 2^{50}, 0]$	$[0, 0, 0, 2^{32}]$	—	$[0, 2^{32}, 2^{32}, 0]^{\dagger}$
5	$[2^{56}, 2, 0, 0]$	—	$[0, 0, 0, 2^{32}]$	—	—
6	$[2^{112}, 4, 0, 0]$	—	$[0, 0, 0, 2^{32}]$	—	$[0, 2^{64}, 2^{96}, 0]^{\dagger}$

† Can be precluded by prepending a serial number to the data string in combination with Padding Method 3

Table 4: Estimated security levels for $n = 128$, $k = 128$, and $m = 64$; the security level is determined by four numbers: off-line block cipher encipherments, known data string/MAC pairs, chosen data string/MAC pairs, and on-line MAC verifications

	key recovery		forgery		
number	brute force	shortcut	guess	xor	birthday
1.1	$[2^{128}, 2, 0, 0]$	—	$[0, 0, 0, 2^{64}]$	$[0, 2, 2^{32}, 0]$	$[0, 2^{64}, 2^{64}, 0]^{\dagger}$
1.2	$[2^{128}, 2, 0, 0]$	—	$[0, 0, 0, 2^{64}]$	—	$[0, 2^{64}, 2^{64}, 0]^{\dagger}$
2.1	$[2^{128}, 2, 0, 0]$	—	$[0, 0, 0, 2^{64}]$	—	$[0, 2^{64}, 2^{64}, 0]^{\dagger}$
5	$[2^{128}, 2, 0, 0]$	—	$[0, 0, 0, 2^{64}]$	—	—

† Can be precluded by prepending a serial number to the data string in combination with Padding Method 3

Table 5: Estimated security levels for $n = 128$, $k = 128$, and $m = 32$; the security level is determined by four numbers: off-line block cipher encipherments, known data string/MAC pairs, chosen data string/MAC pairs, and on-line MAC verifications

	key recovery		forgery		
number	brute force	shortcut	guess	xor	birthday
1.1	$[2^{128}, 4, 0, 0]$	—	$[0, 0, 0, 2^{32}]$	$[0, 2, 2^{48}, 0]$	$[0, 2^{64}, 2^{96}, 0]^{\dagger}$
1.2	$[2^{128}, 4, 0, 0]$	—	$[0, 0, 0, 2^{32}]$	—	$[0, 2^{64}, 2^{96}, 0]^{\dagger}$
2.1	$[2^{128}, 4, 0, 0]$	—	$[0, 0, 0, 2^{32}]$	—	$[0, 2^{64}, 2^{96}, 0]^{\dagger}$
5	$[2^{128}, 4, 0, 0]$	—	$[0, 0, 0, 2^{32}]$	—	—

† Can be precluded by prepending a serial number to the data string in combination with Padding Method 3