ICS 35.040
L 80

中华人民共和国国家标准

GB/T 15852.2—2012

信息技术　安全技术　消息鉴别码　第2部分：采用专用杂凑函数的机制

Information technology—Security techniques—Message Authentication Codes (MACs)—Part 2: Mechanisms using a dedicated hash-function

(ISO/IEC 9797-2:2002, MOD)

2012-12-31 发布　　2013-06-01 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会　发布

目　次

前　言

GB/T 15852《信息技术　安全技术　消息鉴别码》由如下部分组成：

——第1部分：采用分组密码的机制；

——第2部分：采用专用杂凑函数的机制。

本部分是GB/T 15852的第2部分。

本部分按照GB/T 1.1—2009给出的规则起草。

本部分修改采用ISO/IEC 9797-2:2002《信息技术　安全技术　消息鉴别码　第2部分：采用专用杂凑函数的机制》。增加了基于专用杂凑函数WHIRLPOOL的MAC生成方法及例子，更新了附录和参考文献，并将ISO/IEC 9797-2:2002中计算常数的6.3条调整到本部分的第9章。

本部分由全国信息安全标准化技术委员会(SAC/TC 260)提出并归口。

本部分起草单位：中国科学院软件研究所、信息安全国家重点实验室。

本部分主要起草人：吴文玲、张立廷、王鹏、吴双、张文涛、陈华、眭晗。

引　言

本部分规定的第一个 MAC 算法通常被称作 MDx-MAC。它调用一次完整的杂凑函数，但是对其中的轮函数做了细微的修改，把一个密钥加到了轮函数的附加常数上。第二个 MAC 算法通常被称作 HMAC，它调用两次完整的杂凑函数。第三个 MAC 算法是 MDx-MAC 的一个变种，它限制输入长度不大于 256 比特。在只处理较短输入的情况下，它有更好的性能。

本部分规定的三种 MAC 算法采用四种专用杂凑函数；其中，专用杂凑函数 1、2、3 和 4 分别是 ISO/IEC 10118-3:2004 中规定的专用杂凑函数 1、2、3 和 7。使用的专用杂凑函数也可以为国家密码管理部门批准的相应专用杂凑函数。

信息技术　安全技术　消息鉴别码
第2部分：采用专用杂凑函数的机制

1　范围

GB/T 15852的本部分规定了三种采用专用杂凑函数的消息鉴别码算法。这些消息鉴别码算法可用作数据完整性检验，检验数据是否被非授权地改变。同样这些消息鉴别码算法也可用作消息鉴别，保证消息源的合法性。数据完整性和消息鉴别的强度依赖于密钥的长度及其保密性、杂凑函数的算法强度及其输出长度、消息鉴别码的长度和具体的消息鉴别码算法。

本部分适用于任何安全体系结构、进程或应用的安全服务。

2　规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

GB/T 1988—1998　信息技术　信息交换用七位编码字符集(eqv ISO/IEC 646:1991)

ISO/IEC 10118-3:2004　信息技术　安全技术　杂凑函数　第3部分：专用杂凑函数(Information technology—Security techniques—Hash-functions—Part 3:Dedicated hash-functions)

3　术语和定义

下列术语和定义适用于本文件。

3.1

消息鉴别码　message authentication code;MAC

利用对称密码技术，以密钥为参数，由消息导出的数据项。任何持有这一密钥的实体，都可利用消息鉴别码检查消息的完整性和始发者。

3.2

消息鉴别码(MAC)算法密钥　MAC algorithm key

一种用于控制消息鉴别码算法运算的密钥。

3.3

消息鉴别码算法　message authentication code algorithm

消息鉴别码算法简称MAC算法，其输入为密钥和消息，输出为一个固定长度的比特串，满足下面两个性质：

——对于任何密钥和消息，MAC算法都能够快速地计算。

——对于任何固定的密钥，攻击者在没有获得密钥信息的情况下，即使获得了一些(消息，MAC)对，对任何新的消息预测其MAC在计算上是不可行的。

注：一个MAC算法有时被称作一个密码校验函数。计算不可行性依赖于使用者具体的安全要求及其环境。

3.4

输出变换　output transformation

应用在算法中，对迭代操作的输出所进行的变换。

3.5

抗碰撞杂凑函数 collision-resistant hash-function

满足如下性质的杂凑函数：

——寻找两个不同的输入，使得它们的输出相同，在计算上是不可行的。

3.6

消息比特串(数据) data string (data)

杂凑函数的输入比特串。

3.7

杂凑值 hash-code

杂凑函数的输出比特串。

3.8

杂凑函数 hash-function

将任意长消息比特串映射到定长比特串的函数，并且满足如下两个性质：

——对于任何输出，找到它所对应的输入在计算上是不可行的。

——对于任何输入，找到区别于它且和它具有相同输出的输入在计算上是不可行的。

3.9

初始值 initializing value

杂凑函数开始工作时用到的值。

3.10

填充 padding

在消息比特串后面附加额外比特串的操作。

3.11

分组 block

一种定义了长度的比特串。

3.12

轮函数 round-function

将两个长度为 L_1 和 L_2 的比特串映射到一个长度为 L_2 的比特串的函数 $\phi(\cdot,\cdot)$。

注：它被反复地用在杂凑函数中，将长度为 L_1 的比特串和前面长度为 L_2 的输出值相合并。

3.13

字 word

长度为 32 位的比特串。

4 符号和记法

下列符号和记法适用于本部分。

D、D'	将要被输入到 MAC 算法的消息比特串
m	MAC 值的比特长度
q	经过填充和分割操作后，消息比特串 D 的分组个数
$MSB_j(X)$	比特串 X 最左边的 j 比特
$X \oplus Y$	比特串 X 和 Y 的异或值
$X \| Y$	按顺序将比特串 X 和 Y 连接所构成的比特串
$:=$	MAC 算法定义中使用的赋值符号
$\overline{D}$	经过填充的消息比特串
h	杂凑函数

h'	被修改了常数和初始值的杂凑函数 h
$\overline{h}$	简化的杂凑函数 h，没有数据填充和长度附加
H'、H''	长度为 L_2 比特串，在 MAC 算法计算中被用来存储临时结果
IV、IV'、IV_1、IV_2	初始值
k	MAC 算法密钥的比特长度
K	MAC 算法的密钥
K'、K_0、K_1、K_2	MAC 算法 1 和 3 中的导出密钥
$\overline{K}$、$\overline{K_1}$、$\overline{K_2}$	MAC 算法 2 中的导出密钥
$\widetilde{L}$	MAC 算法 3 中表示消息长度的比特串
$OPAD$、$IPAD$	MAC 算法 2 中使用的常数比特串
R、S_0、S_1、S_2	MAC 算法 1 和 3 中，用来导出一系列常数的常数比特串
T_0、T_1、T_2	MAC 算法 1 和 3 中，用来导出子密钥的 128 比特常数
U_0、U_1、U_2	MAC 算法 1 和 3 中，用来导出子密钥的 768 比特常数
ϕ'	使用修改后常数的轮函数
$K_1[i]$	128 比特串 K_1 的第 i 个字，即：$K_1=K_1[0]\|\|K_1[1]\|\|K_1[2]\|\|K_1[3]$
H	杂凑值
L_X	比特串 X 的比特长度
C_i、C_i'	轮函数中用到的常数字
CC_i	专用杂凑函数 4 中用到的常数矩阵
L_1	输入到轮函数 ϕ 的两个比特串中，第一个比特串的比特长度
L_2	输入到轮函数 ϕ 的两个比特串中，第二个比特串的比特长度；轮函数 ϕ 输出值的比特长度；初始值 IV 的比特长度
ϕ	轮函数，即：若 X 和 Y 分别表示长度为 L_1 和 L_2 的比特串，则 $\phi(X,Y)$ 表示将 ϕ 作用到 X 和 Y 所得到的比特串
$+_{32}$	模 2^{32} 加法操作，即：若 A 和 B 是字，那么把 A 和 B 看作是整数的 2 进制表示，计算它们的和再模 2^{32}，所得到的结果在 0 和 $2^{32}-1$ 之间，把它看作为字，记作 $A+_{32}B$

注：$\overline{h}$只能被用来处理长度为 L_1 整数倍的输入比特串。

5 要求

采用本部分 MAC 算法的使用者应当选择：

1) 从第 6、7、8 章中选取一种 MAC 算法；

2) 从 ISO/IEC 10118-3:2004 中的专用杂凑函数 1、2、3 和 7 中选取一个杂凑函数；

3) MAC 的长度 m。

对于 MAC 算法 1 和 2，MAC 的长度 m 应该是一个正整数并且不大于杂凑值长度 L_H。对于 MAC 算法 3，MAC 的长度 m 应该是一个正整数并且不大于杂凑值长度的二分之一，即 $m\leqslant L_H/2$。

对于 MAC 算法 1 和 2，消息比特串 D 的比特长度不大于 $2^{64}-1$；对于 MAC 算法 3，消息比特串 D 的比特长度不大于 256。

对一个具体 MAC 算法、专用杂凑函数、m 值的选择超出了本部分所规定的范围。

注：上述选择将影响 MAC 算法的安全强度，具体请参考附录 B。

生成 MAC 和验证 MAC 应当使用同样的密钥。

6 MAC 算法 1

MAC 算法 1 计算 MAC 值要求调用一次杂凑函数，而且要求修改其中的轮函数常数。

杂凑函数应当从 ISO/IEC 10118-3:2004 中的专用杂凑函数 1、2、3 和 7 中选取。

密钥长度 k 不大于 128 比特。

注：本条款包括 MDx-MAC 的描述[5]。具体来讲，若采用专用杂凑函数 1，MAC 算法 1 也被称作 RIPEMD-160-MAC；若采用专用杂凑函数 2，MAC 算法 1 也被称作 RIPEMD-128-MAC；若采用专用杂凑函数 3，MAC 算法 1 也被称作 SHA-1-MAC；若采用专用杂凑函数 4，MAC 算法 1 也被称作 WHIRLPOOL-MAC。

6.1 MAC 算法 1 的描述

MAC 算法 1 要求如下五步操作：密钥扩展、修改常数和初始值、杂凑操作、输出变换和截断操作。

6.1.1 密钥扩展

若 K 长度小于 128 比特，那么将 K 重复足够多次数，从连接起来的比特串中选取最左边 128 比特作为 128 比特密钥 K'（若 K 的长度恰好为 128 比特，则 $K' := K$），即：

$$K' := MSB_{128}(K \| K \| \cdots \| K)$$

按照如下操作计算子密钥 K_0、K_1 和 K_2：

$$K_0 := \bar{h}(K' \| U_0 \| K')$$

$$K_1 := MSB_{128}(\bar{h}(K' \| U_1 \| K'))$$

$$K_2 := MSB_{128}(\bar{h}(K' \| U_2 \| K'))$$

其中，U_0、U_1 和 U_2 是 768 比特的常数，在第 9 章中有定义。$\bar{h}$ 表示简化的杂凑函数 h，即没有数据填充和长度附加。

注：数据填充和长度附加可以被省略，是因为在这里输入比特串的长度总是 $2L_1$ 比特。

导出的密钥 K_1 被分割成四个字，表示为 $K_1[i](0 \leqslant i \leqslant 3)$，即：$K_1 = K_1[0] \| K_1[1] \| K_1[2] \| K_1[3]$。

从比特串到字的转换，需要规定字节的排列顺序。在这里的转换中，采用 ISO/IEC 10118-3:2004 中对所有专用杂凑函数规定的字节排列顺序。

6.1.2 修改常数和初始值

轮函数中采用的附加常数，被修改为它与 K_1 四个字中的一个进行模 2^{32} 加的结果，比如说：

$$C_0 := C_0 +_{32} K_1[0]$$

在条款 9 中具体规定了 K_1 中的哪个字与哪个常数相加。用 $IV' := K_0$ 取代杂凑函数的初始值 IV，所得的杂凑函数记作 h'，其中的轮函数记作 ϕ'。

6.1.3 杂凑操作

用 D 表示输入到被修改的杂凑函数 h' 中的比特串，即：

$$H' := h'(D)$$

6.1.4 输出变换

再一次应用被修改的轮函数 ϕ'，其中输入的第一个参数为 $K_2 \| (K_2 \oplus T_0) \| (K_2 \oplus T_1) \| (K_2 \oplus T_2)$，第二个参数为 H'（杂凑操作的结果），即：

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

这里 T_0、T_1和 T_2都是长度为 128 的比特串，在条款 9 中对所有专用杂凑函数均有定义。

注：输出变换对应于处理一个额外的数据分组，这个额外的数据分组是在数据填充和长度附加操作之后，由 K_2 导出。

6.1.5 截断操作

取比特串 H''最左边 m 比特，作为 MAC 值，即：

$$\text{MAC} := MSB_m(H'')$$

6.2 MAC 算法 1 的效率

假定填充后的消息比特串包括 q 个分组(这里填充方法由具体的杂凑函数决定)，那么 MAC 算法 1 调用轮函数 $q+7$ 次。

通过预计算 K_0、K_1和 K_2，并且在杂凑函数的应用中用 IV'取代 IV，MAC 算法 1 调用轮函数的次数可以降低到 $q+1$ 次。

处理长的消息比特串时，MAC 算法 1 和相应杂凑函数的性能相当。

7 MAC 算法 2

MAC 算法 2 计算 MAC 值要求调用两次杂凑函数。

杂凑函数应当从 ISO/IEC 10118-3:2004 中的专用杂凑函数 1、2、3 和 7 中选取，并且要求 L_1是 8 的正整数倍，$L_2 \leqslant L_1$。

注：ISO/IEC 10118-3:2004 中的杂凑函数 1、2、3 和 7 满足这些条件。

密钥长度 k 不小于 L_2比特(L_2是杂凑值的比特长度)，不大于 L_1比特(L_1为输入到轮函数的比特串的比特长度)，即：$L_2 \leqslant k \leqslant L_1$。

7.1 MAC 算法 2 的描述

MAC 算法 2 要求如下四步操作：密钥扩展、杂凑操作、输出变换和截断操作。

7.1.1 密钥扩展

在密钥 K 的右侧填充 L_1-k 个 0，所得的长度为 L_1的比特串记作$\overline{K}$。

按照如下的方法，将$\overline{K}$扩展为两个子密钥$\overline{K_1}$和$\overline{K_2}$：

- 将 16 进制的值“36”(二进制表示为“00110110”)重复 $L_1/8$ 次连接起来，所得比特串记作 $IPAD$。然后将$\overline{K}$和比特串 $IPAD$ 相异或，记作$\overline{K_1}$。即：

$$\overline{K_1} := \overline{K} \oplus IPAD$$

- 将 16 进制的值“5C”(二进制表示为“01011100”)重复 $L_1/8$ 次连接起来，所得比特串记作 $OPAD$。然后将$\overline{K}$和比特串 $OPAD$ 相异或，记作$\overline{K_2}$。即：

$$\overline{K_2} := \overline{K} \oplus OPAD$$

7.1.2 杂凑操作

将$\overline{K_1}$和 D 相连接，作为输入到杂凑函数的比特串，即：

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

7.1.3 输出变换

将$\overline{K_2}$和H'相连接，作为输入到杂凑函数的比特串，即：

$$H'' := h(\overline{K_2} \| H')。$$

7.1.4 截断操作

取比特串H''最左边m比特，作为MAC值，即：

$$\text{MAC} := MSB_m(H'')。$$

7.2 MAC算法2的效率

假定填充后的消息比特串包括q个分组(这里填充方法由具体的杂凑函数决定)，那么采用专用杂凑函数1、2和3时，MAC算法2调用轮函数$q+3$次；采用专用杂凑函数4时，MAC算法2调用轮函数$q+4$次。

通过修改杂凑函数代码，MAC算法2调用轮函数的次数可以降低2次。

使用者可以预计算$IV_1 := \phi(\overline{K_1}, IV)$和$IV_2 := \phi(\overline{K_2}, IV)$，并且在第一次调用杂凑函数时用$IV_1$取代$IV$，在输出变换中(第二次调用杂凑函数)用$IV_2$取代$IV$。同时，这也要求对填充方法进行修改。事实上，对杂凑函数实际输入的比特长度少了L_1，这样必须把L_1的值加到L_D上。

处理长的消息比特串时，MAC算法2和相应杂凑函数的性能相当。

8 MAC算法3

注：本条款包括MAC算法1的一个变种，对短的输入(不大于256比特)做了优化。

MAC算法3计算MAC值，要求调用7次简化的轮函数；但是通过预计算，可以降低到调用一次简化的轮函数。

杂凑函数应当从ISO/IEC 10118-3:2004中的专用杂凑函数1、2、3和7中选取。

密钥长度k不大于128比特，MAC值长度m不大于$L_H/2$比特。

8.1 MAC算法3的描述

MAC算法3要求如下五步操作：密钥扩展、修改轮函数的常数、数据填充、应用轮函数和截断操作。

8.1.1 密钥扩展

若K长度小于128比特，那么将K重复足够多次数，从连接起来的比特串中选取最左边128比特作为128比特密钥K'(若K的长度恰好为128比特，则$K' := K$)，即：

$$K' := MSB_{128}(K \| K \| \cdots \| K)$$

按照如下操作计算子密钥K_0、K_1和K_2：

$$K_0 := \overline{h}(K' \| U_0 \| K')$$

$$K_1 := MSB_{128}(\overline{h}(K' \| U_1 \| K'))$$

$$K_2 := MSB_{128}(\overline{h}(K' \| U_2 \| K'))$$

其中，U_0、U_1和U_2是768比特的常数，在条款9中有定义。$\overline{h}$表示简化的杂凑函数h，即没有数据填充和长度附加。

注：数据填充和长度附加可以被省略，是因为在这里输入比特串的长度总是$2L_1$比特。

导出的密钥 K_1 被分割成四个字，表示为 $K_1[i](0 \leqslant i \leqslant 3)$，即：$K_1 = K_1[0] \| K_1[1] \| K_1[2] \| K_1[3]$。

从比特串到字的转换，需要规定字节的排列顺序。在这里的转换中，采用 ISO/IEC 10118-3:2004 中对所有专用杂凑函数规定的字节排列顺序。

8.1.2 修改轮函数的常数

轮函数中采用的附加常数，被修改为它与 K_1 四个字中的一个进行模 2^{32} 加的结果，比如说：

$$C_0 := C_0 +_{32} K_1[0]$$

在第 9 章中具体规定了 K_1 中的哪个字与哪个常数相加。用 $IV' := K_0$ 取代杂凑函数的初始值 IV，所得的轮函数记作 ϕ'。

8.1.3 数据填充

对原始消息填充的比特串只用来计算 MAC，所以这些填充比特串（如果有）不必随原始消息存储或发送。MAC 的验证者应当知道填充比特串是否已经被存储或发送。

对要输入到 MAC 算法的消息比特串 D，在其右侧填充尽可能少（可能没有）的“0”以使得填充后比特串 $\overline{D}$ 的长度是 256 比特。

注：如果消息比特串 D 是空串，那么规定填充后的比特串 $\overline{D}$ 为 256 个“0”。

8.1.4 应用轮函数

消息比特串 D 的长度记作 L_D，其二进制表示记作 $\widetilde{L}$。在 $\widetilde{L}$ 最左边填充足够少的“0”使得 $\widetilde{L}$ 的长度为 128 比特，$\widetilde{L}$ 最右边的比特和 L_D 最低位相对应。

将 K_2、$\overline{D}$ 和 K_2 与 $\widetilde{L}$ 的异或值相连接，作为轮函数 ϕ'（使用修改过的常数）的输入，即：

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

8.1.5 截断操作

取比特串 H' 最左边 m 比特，作为 MAC 值，即：

$$\text{MAC} := MSB_m(H')。$$

8.2 MAC 算法 3 的效率

MAC 算法 3 需要调用 7 次轮函数，通过预计算 K_0、K_1 和 K_2，可以降低到一次。

9 常数的计算

本条款中规定的常数，将被用在 MAC 算法 1 和条款 8 的 MAC 算法 3 中。

比特串 T_i 和 U_i 是 MAC 算法中固定的元素，它们通过杂凑函数计算得到（只计算一次），并且在四个专用杂凑函数中各不相同。

128 比特的 T_i 和 768 比特的 U_i 按照如下的方法定义：

$$T_i := MSB_{128}(\overline{h}(S_i \| R)), \qquad i=0,1,2$$

$$U_i := T_i \| T_{i+1} \| T_{i+2} \| T_i \| T_{i+1} \| T_{i+2}, \qquad i=0,1,2$$

其中下标的加法是模 3 加。$R =$ "$ab\cdots yzAB\cdots YZ01\cdots 89$" 是 496 比特的常数，$S_0$、$S_1$ 和 S_2 都是 16 比特的常数，其中 S_i 通过重复两次数字 i 的 16 进制 ASCII 编码得到（比如说，S_1 的表示为 3131）。R 和 S_i 都采用 ASCII 编码，ASCII 编码等同于 GB/T 1988—1998 所使用的编码。

对于所有的常数 C_i、C'_i和所有的字 $K_1[i]$，最高位和最左边的比特相对应。常数 C_i 和 C'_i用 16 进制表示。

9.1 专用杂凑函数 1

专用杂凑函数 1 中的 128 比特常数 T_i 定义如下：(用 16 进制表示)

$$T_0 = 1CC7086A046AFA22353AE88F3D3DACEB$$

$$T_1 = E3FA02710E491D851151CC34E4718D41$$

$$T_2 = 93987557C07B8102BA592949EB638F37$$

专用杂凑函数 1 的轮函数中用到两个常数字序列 $C_0, C_1, \cdots, C_{79}$ 和 $C'_0, C'_1, \cdots, C'_{79}$，它们定义如下：

$$C_i = K_1[0] +_{32} 00000000, (0 \leqslant i \leqslant 15),$$

$$C_i = K_1[1] +_{32} 5A827999, (16 \leqslant i \leqslant 31),$$

$$C_i = K_1[2] +_{32} 6ED9EBA1, (32 \leqslant i \leqslant 47),$$

$$C_i = K_1[3] +_{32} 8F1BBCDC, (48 \leqslant i \leqslant 63),$$

$$C_i = K_1[0] +_{32} A953FD4E, (64 \leqslant i \leqslant 79),$$

$$C'_i = K_1[1] +_{32} 50A28BE6, (0 \leqslant i \leqslant 15),$$

$$C'_i = K_1[2] +_{32} 5C4DD124, (16 \leqslant i \leqslant 31),$$

$$C'_i = K_1[3] +_{32} 6D703EF3, (32 \leqslant i \leqslant 47),$$

$$C'_i = K_1[0] +_{32} 7A6D76E9, (48 \leqslant i \leqslant 63),$$

$$C'_i = K_1[1] +_{32} 00000000, (64 \leqslant i \leqslant 79)$$

9.2 专用杂凑函数 2

专用杂凑函数 2 中的 128 比特常数 T_i 定义如下：(用 16 进制表示)

$$T_0 = FD7EC18964C36D53FC18C31B72112AAC$$

$$T_1 = 2538B78EC0E273949EE4C4457A77525C$$

$$T_2 = F5C93ED85BD65F609A7EB182A85BA181$$

专用杂凑函数 2 的轮函数中用到两个常数字序列 $C_0, C_1, \cdots, C_{63}$ 和 $C'_0, C'_1, \cdots, C'_{63}$，它们定义如下：

$$C_i = K_1[0] +_{32} 00000000, (0 \leqslant i \leqslant 15),$$

$$C_i = K_1[1] +_{32} 5A827999, (16 \leqslant i \leqslant 31),$$

$$C_i = K_1[2] +_{32} 6ED9EBA1, (32 \leqslant i \leqslant 47),$$

$$C_i = K_1[3] +_{32} 8F1BBCDC, (48 \leqslant i \leqslant 63),$$

$$C'_i = K_1[0] +_{32} 50A28BE6, (0 \leqslant i \leqslant 15),$$

$$C'_i = K_1[1] +_{32} 5C4DD124, (16 \leqslant i \leqslant 31),$$

$$C'_i = K_1[2] +_{32} 6D703EF3, (32 \leqslant i \leqslant 47),$$

$$C'_i = K_1[3] +_{32} 00000000, (48 \leqslant i \leqslant 63)$$

9.3 专用杂凑函数 3

专用杂凑函数 3 中的 128 比特常数 T_i 定义如下：(用 16 进制表示)

$$T_0 = 1D4CA39FA40417E2AE5A77B49067BBCC$$

$$T_1 = 9318AFEF5D5A5B46EFCA6BEC0E138940$$

$$T_2 = 4544209656E14F97005DAC76868E97A3$$

专用杂凑函数 3 的轮函数中用到一个常数字序列 $C_0, C_1, \cdots, C_{79}$，它定义如下：

$$C_i = K_1[0] +_{32} 5A827999, (0 \leqslant i \leqslant 19),$$

$$C_i = K_1[1] +_{32} 6ED9EBA1, (20 \leqslant i \leqslant 39),$$

$$C_i = K_1[2] +_{32} 8F1BBCDC, (40 \leqslant i \leqslant 59),$$
$$C_i = K_1[3] +_{32} CA62C1D6, (60 \leqslant i \leqslant 79)$$

9.4 专用杂凑函数 4

专用杂凑函数 4 中的 128 比特常数 T_i 定义如下：(用 16 进制表示)

T_0 = 36129D71469F205824E118E1F922295A

T_1 = 19A934B16952C37951C31AA9699FD5A4

T_2 = 313D9B56799988C1A3639FB2BDFEF274

专用杂凑函数 4 的轮函数中用到一个常数矩阵序列 $CC_1, CC_2, \cdots, CC_{10}$，它们定义如下：

$$CC_1 = \begin{bmatrix} 1823C6E8 & 87B8014F \\ K_1[0] & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \end{bmatrix}, \quad CC_2 = \begin{bmatrix} 36A6D2F5 & 796F9152 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ K_1[1] & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \end{bmatrix},$$

$$CC_3 = \begin{bmatrix} 60BC9B8E & A30C7B35 \\ K_1[2] & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \end{bmatrix}, CC_4 = \begin{bmatrix} 1DE0D7C2 & 2E4BFE57 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ K_1[3] & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \end{bmatrix},$$

$$CC_5 = \begin{bmatrix} 157737E5 & 9FF04ADA \\ K_1[0] & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \end{bmatrix}, CC_6 = \begin{bmatrix} 58C9290A & B1A06B85 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ K_1[1] & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \end{bmatrix},$$

$$CC_7 = \begin{bmatrix} BD5D10F4 & CB3E0567 \\ K_1[2] & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \end{bmatrix}, CC_8 = \begin{bmatrix} E427418B & A77D95D8 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ K_1[3] & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \end{bmatrix},$$

$$CC_9=\begin{bmatrix} \text{FBEE7C66} & \text{DD17479E} \\ K_1[0] & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \end{bmatrix}, CC_{10}=\begin{bmatrix} \text{CA2DBF07} & \text{AD2A8333} \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \\ K_1[1] & 00000000 \\ 00000000 & 00000000 \\ 00000000 & 00000000 \end{bmatrix}。$$

附 录 A
（资料性附录）
使用 MAC 算法生成 MAC 的示例

A.1 概述

本附录提供了使用 MAC 算法 1、2 和 3 生成 MAC 的过程示例，采用四种专用杂凑函数。其中，专用杂凑函数 1、2、3 和 4 分别是 ISO/IEC 10118-3:2004 中规定的专用杂凑函数 1、2、3 和 7。每个杂凑函数值的计算有九个示例。表 A.1 包含了序号为 1 至 9 的输入比特串。在整个附录中，对消息比特串采用 ASCII 编码，ASCII 编码等同于 GB/T 1988—1998 所使用的编码。

两个 128 比特的密钥如下：

密钥 1=00112233445566778899AABBCCDDEEFF

密钥 2=0123456789ABCDEFFEDCBA9876543210

表 A.1 用于测试的输入数据

序号	输入比特串
1	“”(空比特串)
2	“a”
3	“abc”
4	“message digest”
5	“abcdefghijklmnopqrstuvwxyz”
6	“abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq”
7	“ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789”
8	“1234567890”重复 8 次所得的长为 80 的字符串
9	“a”重复 1 000 000 次得到 1 兆长的字符串

A.2 MAC 算法 1

在这一部分的示例中，选取 $m=L_2/2$。具体来讲，在专用杂凑函数 1 和 3 中，$m=80$；在专用杂凑函数 2 中，$m=64$；在专用杂凑函数 4 中，$m=256$。

A.2.1 专用杂凑函数 1

密钥 1:00112233445566778899AABBCCDDEEFF	
序号	MAC 值
1	B7F4508111EB8C3B5229C6AED406DE9ECA640133
2	BC78F55933BCEB1EE85A906F9E18374F23E310F9
3	6300DC20E97A5AA29DB9C7D607D23D126FA36863

表（续）

密钥 1:00112233445566778899AABBCCDDEEFF	
序号	MAC 值
4	3A2AC89B78EEAB8759F5112BCAD4CD405EEB5D35
5	16DC174925BBC27E0C93D426C346846F97F8BC69
6	E062210BA5C9C94737BF3A6E85B3B5664FBD1D4E
7	9B462D5CBDAE1485FFE10BC001EF9E3AF6D128B5
8	88E73A01A1DE36C92D6F9E41F7278D407B4A4CCD
9	E7B128E4A1842B750F1E61A486C867C4887A4B21

密钥 2:0123456789ABCDEFFEDCBA9876543210	
序号	MAC 值
1	B45D6CA84CFB9020E0D5ABA2A7609D3D81F3F57F
2	8844375992037D1BCD0D118EE548D70C3F19CBBB
3	917C59B8AC7FC19DC25BEF82766412FA16BBC6A7
4	E0737CC7976D8F424390CB8798D623D751AFE15A
5	D57FAE83687-718EFA4BD4A5F2F322A179A8735E
6	42B20D4C8FD5E8672760CF83C0478D7BF8021404
7	42B20D4C8FD5E8672760CF83C0478D7BF8021404
8	10441DF4F68CE8815818DC0FB370ABF87BCA4464
9	E06AD21D2AF04DD4217AB03B1A578F036997D01A

A.2.2 专用杂凑函数 2

密钥 1:00112233445566778899AABBCCDDEEFF	
序号	MAC 值
1	A47A64E9EDE0741B3FDDE33E5C1C6D78
2	51355051852FDC79FB228EAC905633AD
3	D83940DAFFBD4CBBE6BA30A6F9E63F5F
4	1A7CFE2BB26E973E213C1CB96FA4C2EF
5	798AEAC6046B31907C197BD68E59D376
6	0B8E1D4A571F32657189322A1F2F4A53
7	B814730F482300C6E474FD255A66D680
8	9060A30758EBE3368D939AC168F1A9FD
9	20763FDEDF01E56FF5756954302C7DE0

密钥 2:0123456789ABCDEFFEDCBA9876543210	
序号	MAC 值
1	35FA3AC39F50F2A4E3FFC7AF5776B4EB
2	A89E25E6796747B630A2A00B802EA53E
3	66339027A36608EBD932DD551616E7B2
4	1F8779BAD84B50373931211A2761EAD3
5	31BF5B5B7ABAC2567DC0E02F1C3A25D7
6	B5B8BA3B8EA895FBC83CB7588FBD2656
7	8D27BBEC257C848D5CF375EB5EDA4CC7
8	B40B5BF6727DE90B26F770850F059C89
9	76C7BC831B0BCE593DFD44E8E054A373

A.2.3 专用杂凑函数 3

密钥 1:00112233445566778899AABBCCDDEEFF	
序号	MAC 值
1	C8A8B3C75E6CE7C6C4F79CC19853CCD54ABCB079
2	8DD9AE643BF10BBB7B978EF13EE6C0F480618FB0
3	A738B26A8BD318184E76707A99CAE14C670B9711
4	1EBFE413E55D6B288A2BD01D294A21FD8D4B20BF
5	0CE7BF40A73D977AB4999CF3A9BD1C5B3DC442E9
6	12A6823CC181294F95109073A6AA0C8961B14386
7	9369EE4A043AF1CA6E078D0B8A9CE5C1545440BA
8	B00D37D70A84B762FC0A8A9BC1B15F0E517B5EDF
9	DDDF44613E8559D12C150D022D5FE33F9E0FBACE

密钥 2:0123456789ABCDEFFEDCBA9876543210	
序号	MAC 值
1	C3A5ECD1E715C7272CFE78BC278086587B040422
2	D5D50FFA7EFDF1B17E96E2EC14DBC4412F7B771F
3	01BFDD568008D412158F5B0C90AE2730DCFB77FB
4	9982E0EE91DB89AE7E7618AD1D649BA43406DBDD
5	ACD04E1004FCE53DECA9EE7AB95DAF97B7C44AA8
6	FADF62DCE789E86E60756AA819EF62C8E5C25E94
7	46DB9A49FB4976D007B14B1574843D019CA99445
8	4EF5BED3E816C530B23F491583C038596BB76FDB
9	BAC6BE6BE6153FECE2891F9DA03824DD4D535D19

A.2.4 专用杂凑函数 4

密钥 1:00112233445566778899AABBCCDDEEFF	
序号	MAC 值
1	064F757F9F4A5E84CC0A859CD5A3658C3B0BD0C2E8FC8955724E3E0022DF1F05
2	05AF97181F1C367FD8A4D0B7803A79C96EA8EB64829CAF8CBB68CA6302342CDE
3	A7D9D03F712C5942FBA478B7CA18FA567E506E60A5B121520BDE1D22E7C0993B
4	40C93B6C941A84EB4589EBAF0EF3EB68D056DA8D854795ACE567D5A7C6FD52C7
5	6893767C115A21DCA5828BC3F9640DE7473E22C2572657EF47327E9C238FE0C5
6	3F3A93D580FBCE562DE0AD7AF321E5ED7B8E81816C300B7F5FC21DF80EEE7F71
7	5FC693ED3C3763AF071FA3A02B79E0D5D35041C709D9ECA95662AF2DCC9A0090
8	D4E2D986D5BC4CFFDAF05C6D1C0E433910C867C06D47A5A9B2736E8AAB490150
9	E155B811AAD52FA647452BAC9F97F9B6134634605920C23CBE0E843C931BA2DA

密钥 2:0123456789ABCDEFFEDCBA9876543210	
序号	MAC 值
1	AC20EA6E726BD180EB0BA5DFF9761F459AFB67F803496172FFDC6D1482D7E3FB
2	A583B2B736F7450FF3E83CDA1A12647D4A44DBA64FECCBE7A05F526AA63BCC6C
3	E0B2DE807F2506A2FA5F6E6AEDEDF348690BF6A4B58EAA4767D7EFD871051912
4	46C4DF153EBA0BD231919EC2D26F611B66B635836F5235AAA8D2BB240329F657
5	2D174A353C4FD84177E43F67B3CD685D3549BD038EEFD5A496F3855F73110CEC
6	06B49ABF50036613798815119D94A37B26B975C53D8A753037FBABF6F6774422
7	FD74C324D3C4D5001797ACD458995FEB4C50F3D5502D778895EE4E6353BF3586
8	00BB8AB26F991C10EBD0789467C67EC4988E51ED78A9A9596C36E9817D63B7C1
9	B5B781DC131DF455727A5DDADA1F4BC3402E74865D28917D0B31E9AF61D0CE07

A.3 MAC 算法 2

在这一部分的示例中，选取 $m=L_2/2$。具体来讲，在专用杂凑函数 1 和 3 中，$m=80$；在专用杂凑函数 2 中，$m=64$；在专用杂凑函数 4 中，$m=256$。

A.3.1 专用杂凑函数 1

密钥 1:00112233445566778899AABBCCDDEEFF	
序号	MAC 值
1	9EBEA41FBC24CD90BF2ECFD5B8C8CC8181D3FCAE
2	75CB722C50024C0E8A7A0DBA7D5C36B86D9D1DD5
3	5B48C1749DDED71EDFE0ADE2B944E808E4A65820
4	F9033064567F541235C3944EE95CB476055985D1
5	B37885405B71E025AF0CB574021A562A62733628
6	5C6429B982C8054B5B3348A0D7D2CE24D7032BC1
7	B0A4A451D0926855E52428E16D1FEAA241C4DD9B
8	1CCEEC5122F08A76EBCD8E3DE88610D942D8A5F6
9	45D61908BFF6039E6DE3C037FDCE6191F19F6410

密钥 2:0123456789ABCDEFFEDCBA9876543210	
序号	MAC 值
1	2FDE5DAF7050D14E6D7ACD2254D17FA3A8CBFCDD
2	239C4020610429A8662BF81A2CAAEA47F8EA0A44
3	89EFFB9F5A6BCEAE3C65D0C9803F3464E5E9E349
4	F5FC87FD5702F5D4E7BB634DA4CB4B41CD505B6C
5	5686C00F69E6C868732C67402AA107CEAB513439
6	525EC4893A221EFD9B6DD351059B40C05B4CE2D3
7	B975ED3893FC8D535376EF49211E2E6B1BB30B90
8	BC201FFA581357C271DAE25104167F3DCC97BADC
9	95A875A1D64D55E677D8E4455E1445E7E940F758

A.3.2 专用杂凑函数 2

密钥 1:00112233445566778899AABBCCDDEEFF	
序号	MAC 值
1	AD9DB2C1E22AF9AB5CA9DBE5A86F67DC
2	3BF448C762DE00BCFA0310B11C0BDE4C
3	F34EC0945F02B70B8603F89E1CE4C78C
4	E8503Q8AEC2289D82AA0D8D445A06BDD
5	EE880B735CE3126065DE1699CC136199
6	794DAF2E3BDEEA2538638A5CED154434
7	3A06EEF165B23625247800BE23E232B6
8	9A4F0159C0952DA43A8D466D46B0AF58
9	19B1B3AF333B894DD86D09427116D0AD

密钥 2:0123456789ABCDEFFEDCBA9876543210	
序号	MAC 值
1	8931EEEE56A6B257FD1AB5418183D826
2	DBBCF169EA7419D5BA7BD8EB3673FF2D
3	2C4CD07D3162D6A0E338004D6B6FBC9A
4	75BFB25888F4BB77C77AE83AD0817447
5	B1B5DC0FCB7258758855DD1840FCDCE4
6	670D0F7A697B18F1A8AB7D2A2A00DBC1
7	54E315FDB34A61C0475392E5C7852998
8	AD04354D8AA2A623E72E3594EE3535C0
9	6F9B1C0FC06753618D6DB4B007733795

A.3.3 专用杂凑函数 3

密钥 1:00112233445566778899AABBCCDDEEFF	
序号	MAC 值
1	86C2962E58B3498A2608935AF7726311F2BFB538
2	0497FF21DAE3251DA0ED2F47F5A3B74ABA6B2560
3	6EE2A25F943E3F3EC05225FBB86BA73E2E5D51D2
4	CD4C0D1328DC4A8DC2801001B129AEFC6E0CF9CE
5	89ECE303FAD1E4313950CC3B008CB239B5B85844
6	9DF741057D075D3C4E1533E38A5FF469647194B4
7	188A58390A6EF9827035B81CDF1B5049211F0EE5
8	98A98D6A81FD361030856D2C19742AD8DBC468E7
9	D2986310BA18A78786534882F9C6BCBF06CCE9E3

密钥 2:0123456789ABCDEFFEDCBA9876543210	
序号	MAC 值
1	2739B6BE63F539EB70FE250346F6382A2DFA345F
2	A0C2711A6B1DA4CD8F85EF1E6FF7BF70B412B477
3	18F570E864FF903D2773D53C2E114E1A62152953
4	A80845A89BA15E941A2457084BC431F3E47759E1
5	14143EA1057B02D20C0157216190A006E30F3D41
6	DAB4B41BA639B4715889406FE18E0C037017E063
7	AEAEA5415B4F266CB15CBEB844E56AEC2DABAD6D
8	3DBA11471EB4FCCF21BAEB0BFF7E20150132C6CF
9	3BB917B8BD8560E89FF9054FBE096CBACA109D5F

A.3.4 专用杂凑函数 4

密钥 1:00112233445566778899AABBCCDDEEFF	
序号	MAC 值
1	5A77B599D2DB9B6B8C8E5112DD5F0B88719D60A4866688C2DFF624A6EA4ADB62
2	177B98F9F215046E640B8EFE3E723C4E7233C5E745B72DE9381D6A3F47E30F95
3	F92ECF9FB82E39ECB2D3EC8CFEF76317A8C4E6F835BD4994D4D156B68F640D37
4	078067C14C1E393011BFE58E1E94F03F9062DA01378760A65F7BE1FF041B8087
5	05411A95D2A5CDA0CB4A7339A70E62FF790D945F25963F1595E39486BAD88B2F
6	8E2A8C15E9611E575BF67165B38B04259A30C8C15F9DE72997391B32575D9C78
7	9A7D93D28BA451CDE57570C1CC41E943D288F3FD112C7E3222185F2163AE9328
8	A3676A07D9E79CABDAA1DA6EAB3FBAD128114F4D7E00050AB7167400203585B6
9	521EA57548F1068EC0364330ABEEAC859E008D976323B1BA13ECFB405E0909EB

密钥 2:0123456789ABCDEFFEDCBA9876543210	
序号	MAC 值
1	FDB6120AFEDAEB87A4DDC952FE02C1ECB17DDD6647D0FAB67194CAA506EED1DB
2	8B738011A43BD81363C38B941E81975BC2562EC9185B70B5503D34FEA89B0E3B
3	C97109474261CEDB4FE524CE8319BD1E4FAD2DCA5434840030238EB26812644D
4	A320497D440E9452846B80EFD4578628ACD969D64A6EC42EF350F05BE6F604E8
5	E1C734A8E6301FD270655F5E6DACCE5115083D3DA974D41182C219F74F357E48
6	66E060BF156AEC454058E4D4A88A0DA88FAD6D118D5B731060FA0BB68B673DDB
7	608FB970FD10D1BBCEAEE1FA02E44C062F1711A214E2594BE57A71FCC419042F
8	D3B314AD10D07CC45708D35526B165A89B5AE596D24ABEACFCD3C0EF2DCCF196
9	024F0B3B7A403417B8191F8383DFFE55F23F5B1A29E3FC24BB29097E294FE798

A.4 MAC 算法 3

在这一部分的示例中，选取 $m=L_2/2$。具体来讲，在专用杂凑函数 1 和 3 中，$m=80$；在专用杂凑函数 2 中，$m=64$；在专用杂凑函数 4 中，$m=256$。

A.4.1 专用杂凑函数 1

密钥 1:00112233445566778899AABBCCDDEEFF	
序号	MAC 值
1	6606EF2D3BBD010F516C65372C3CF0ACF111B3F7
2	F0BC0C81307E17A71F4C40AE0B2AC39FCB23CE12
3	7720FD23925B854F963E8812573CD86EBA61EB66
4	2683D6CE053BA0420E76130EAE2367734B7D2D53
5	DE532D156CB312464BB6147E99470C471D91F1C6

密钥 2:0123456789ABCDEFFEDCBA9876543210	
序号	MAC 值
1	4BD390E9EC460AD4866CCB32D091AFBF73E5B6DA
2	CD2847BAB4636C9BCEADCF3D187122A9199DA670
3	15C3910C42638E5EE6DEBD506BD8C4DB94713A3A
4	04148DCB47728E3E57B836A66043D5145879796E
5	829A24010704DBD0EE34A6D607F7B34829E04E95

A.4.2 专用杂凑函数 2

密钥 1:00112233445566778899AABBCCDDEEFF	
序号	MAC 值
1	AEB2C45F13C0C6F5F10BE2F1E3E9C322
2	16874D0E17E4F1C290DD749CCEFF7834
3	A289AA06AEB8FC99B989C377BAADD9D4
4	0D80DB68BBF99442DC3D6B83D038DEF3
5	11DC4A6BD375C64F78BB78AD265ED7CD

密钥 2:0123456789ABCDEFFEDCBA9876543210	
序号	MAC 值
1	7248481816B8D3AF29F5C002FF769EA5
2	DFE1E36CE9792476E0889F1BECEF18A9
3	9B4F1D21320F4A327F023947554BFC3B
4	3D2D658D0196E4EE9F42ADA50DFCFA6F
5	0A34452D9DA70C70183DFFDDB8EEC056

A.4.3 专用杂凑函数 3

密钥 1:00112233445566778899AABBCCDDEEFF	
序号	MAC 值
1	708F4A226CDE708820643CBEEFDCBE6CB36FB269
2	EAB87BE709D1E5CB62C7489C2B1407130B772760
3	C1BD6F9C908132FEF5187CBE681B42A8C785FBF6
4	F34DEB241D46C6448D67ACE6B8CD4DF00DA23EBC
5	669DED2BD6A1AE0BCFF7D3B74494C1D8161FA0D8

密钥 2:0123456789ABCDEFFEDCBA9876543210	
序号	MAC 值
1	EAF6F9DDBAFD299320FF0FC8E02E2BE62879F341
2	2AAE9DE0A555E7CD7383C27506A467B8DF4E3A33
3	FE6031710329D12090F73F55CFFCC6215F9BEAE9
4	0CCDD9DAB6B0126800EC1CC7A02656E12EDEA42C
5	ABDBC8AAAE4A8CE734432188740A149BDF2D215F

A.4.4 专用杂凑函数 4

密钥 1:00112233445566778899AABBCCDDEEFF	
序号	MAC 值
1	35E68BCFCD5548A09F6A1615B84BEE9AAE35D286BD948BFD7EC1132A8D462C88
2	72E8DA475B0F5C97F71D7A98FB3E0D4E1032AF8080F3BD793EC034B06E619067
3	45AD62CA5A90E3AFD20B645AAC8D77614DB847790867F348D1732BB9BA816C1E
4	039731A1305C30B2F443D403F40B55C6B3B16B5B1B20B60B5942B01E16D0ADEA
5	7F2E3B78CAB48C93E6E7C33BD52B2911C3FBB5BB3F9D40242BB5861D70C1D29C

密钥 2:0123456789ABCDEFFEDCBA9876543210	
序号	MAC 值
1	BB52A0272197E3C112A502A994A12B20CB257C7C4F00D134E9B85E92CD280907
2	DB4C2DC7512E00D835FAF9680F855EAE1379B6A380A8C53F2507F5A5AF0A447C
3	3D75654FC093B5318C05455A212416FD3A4C82F58468C3327C5C8109F7973FBA
4	BE3D19D1E4D6A0F644742A6EAB9D371447D77B8BAE26AE63D4797C5CBBB10E71
5	84EE551EF07D23E2E043061E2C1E6D9ACBADDDAFAF8FD6FC98A384F6FEEB2B0E

附 录 B
（资料性附录）
MAC 算法的安全性分析

本附录讨论了本部分中 MAC 算法的安全强度。它的目标是协助本部分的使用者选择合适的 MAC 算法。

本附录中，$MAC_K(D)$表示用密钥为 K 的 MAC 算法对消息 D 进行计算所得到的 MAC。

为了确定 MAC 算法的安全强度，本附录考虑了如下两种攻击策略：

1) 伪造攻击：这种攻击是在没有密钥 K 的情况下，对消息 D 预测 $MAC_K(D)$。如果攻击者能够对一个消息成功预测其 MAC，那么称他有能力“伪造”。实际的攻击经常要求一个伪造是可验证的，也就是说，事先以接近于 1 的概率确认伪造的 MAC 是正确的。而且，在许多应用中，消息有特定的格式，这就意味着对消息 D 有额外限制。

2) 密钥恢复攻击：这种攻击根据大量的（消息，MAC）对找到 MAC 算法的密钥 K。密钥恢复攻击比伪造攻击更强大，因为它一旦成功就可以进行任意地伪造。

一个攻击的可行性依赖于攻击者已知和选择的（消息，MAC）对数目以及离线加密的次数。

对 MAC 算法可能的攻击描述如下，但是这里并不保证列举了所有的攻击。前两种攻击是一般性的，也就是说，它们对任何 MAC 算法都有效。第三种适用于任何迭代的 MAC 算法（更多信息请参阅[5]）。

- **猜测 MAC**。这种伪造是不可验证的，成功概率为 $\max(1/2^m, 1/2^k)$。这种攻击适用于所有的 MAC 算法，只有合适地选择 m 和 k 才能够抵抗这种攻击。
- **密钥穷搜索**。这种攻击需要运行平均 2^{k-1} 次 MAC 算法，并且需要 k/m 对（消息，MAC）以唯一确定密钥。同样这种攻击适用于所有 MAC 算法，合适地选择 k 能够抵抗这种攻击。另外，MAC 算法使用者也可以阻止攻击者获得 k/m 对（消息，MAC）以抵抗这种攻击。比如说，若 $k=128$，$m=64$，那么对于给定的（消息，MAC）对，大约有 2^{64} 个密钥和其对应；如果每次使用 MAC 算法后都改变密钥，那么密钥穷搜索攻击并不比猜测 MAC 攻击更有效。
- **生日攻击**[5]。如果攻击者获得足够数目的（消息，MAC）对，他就能够找到这样两个消息 D 和 D' 满足：$MAC_K(D)=MAC_K(D')$ 并且两次输出变换的输入值在两次 MAC 计算中是相等的；这被称作内部碰撞。如果消息 D 和 D' 构成内部碰撞，那么对任意的比特串 Y 都有 $MAC_K(D \| Y)=MAC_K(D' \| Y)$。这就构成了一种伪造，当攻击者得到比特串 $D \| Y$ 的 MAC 时，就能够预测比特串 $D' \| Y$ 的 MAC。这种伪造依赖于消息的特殊格式，可能对许多应用没有威胁；但是，这种攻击的扩展版本在消息格式方面有更大的灵活性。这种攻击需要选择一个消息比特串，收集大约 $2^{n/2}$ 对已知（消息，MAC）和 2^{n-m} 对选择（消息，MAC），其中 n 是中间状态的比特长度。

通过以下方式可以避免生日攻击：在要处理的消息前面加上一个序列号消息块，使得 MAC 算法是带状态的。这就要求在 MAC 算法实现中要保证在一个密钥周期内，每个序列号在 MAC 计算过程中只用一次。这种要求并不是在所有环境下都可行。

- **捷径密钥恢复**。基于内部碰撞的密钥恢复攻击适用于某些 MAC 算法，但目前还没有关于本部分中 MAC 算法的捷径密钥恢复攻击。
- **侧信道攻击**。针对 RIPEMD-160 和 SHA-1 算法中异或、模加和模乘运算的特点，可以使用差分能量分析的方法恢复 MAC 算法 2 的全部密钥[6]。

安全证明

若如下的假定成立，那么 MAC 算法 1 被证明是安全的[4]：

- 使用密钥为初始值 IV 和附加常数的轮函数 ϕ 是一个伪随机函数。

注：一个伪随机函数使用一个密钥，对于不知道密钥的敌手，它的表现和一个随机函数相仿（也就是说，伪随机函数和随机函数很难区别）。

若如下的假定成立，那么 MAC 算法 2 被证明是安全的[3]：

- 当 IV 保密的时候，杂凑函数 h 是抗碰撞的。
- 使用密钥为初始值 IV 的轮函数 ϕ 是一个强 MAC 算法（也就是说，很难预测它的输出）。
- 密钥 $\overline{K_1}$、$\overline{K_2}$ 和随机密钥不可区分；相当于要求使用密钥 IV 的轮函数 ϕ 是“弱”伪随机函数（“弱”源于敌手不能直接得到 $\overline{K_1}$ 和 $\overline{K_2}$）。

进一步的证明指出，只要轮函数 ϕ 是伪随机函数，即可保证 MAC 算法 2 是伪随机函数[7]。另外，当轮函数 ϕ 具备不可延展和不可预测性质的时候，MAC 算法 2 被证明是不可伪造的[8]。

其他研究表明，当采用弱的杂凑函数的时候，MAC 算法 2 的安全性有所降低[9~13]。

MAC 算法 3 的安全性假设和 MAC 算法 1 和 2 中对轮函数 ϕ 的假设相似。

参 考 文 献

[1] ISO 16609:2004 Banking-Requirements for message authentication using symmetric techniques.

[2] ISO/IEC 10181-6:1996 Information technology-Open Systems Interconnection-Security frameworks for open systems:Integrity framework.

[3] M. Bellare, R. Canetti, H. Krawczyk, "Keying hash functions for message authencication," Advances in Cryptology, Proceedings Crypto'96, LNCS 1109, N. Koblitz, Ed. , Springer-Verlag, 1996, pp. 1-15.

[4] M. Bellare, R. Canetti, H. Krawczyk, "Pseudorandom functions revisited: The cascade construction and its concrete security," Proc. 37th Annual Symposium on the Foundations of Computer Science, IEEE, 1996, pp. 514-523. Full version via http://www-cse. ucsd. edu/users/mihir.

[5] B. Preneel, P. C. van Oorschot, "MDx-MAC and building fast MACs from hash functions," Advances in Cryptology, Proceedings Crypto'95, LNCS 963, D. Coppersimith, Ed. , Springer-Verlag, 1995, pp. 1-14.

[6] K. Lemke, K. Schramm, C. Paar, "DPA on n-bit sized boolean and arithmetic operations and its application to IDEA, RC6, and the HMAC-construction," Proceedings CHES'04, LNCS 3156, M. Joye and J. J. Quisquater Eds. , Springer-Verlag, 2004, pp. 205-219.

[7] M. Bellare, "New proofs for NMAC and HMAC: security without collision-resistance," Advances in Cryptology, Proceedings Crypto" 06, LNCS 4117, Dwork Ed. , Springer-Verlag, 2006, pp. 602-619.

[8] M. Fischlin" Security of NMAC and HMAC based on non-malleability," Proceedings CT-RSA'08, LNCS 4964, T. Malkin, Ed, Springer-Verlag, 2008, pp. 138-154.

[9] S. Contini, Yiqun Lisa Yin" Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions," Proceedings ASIACRYPT'08, LNCS 4284, X. Lai and K. Chen, Eds, Springer-Verlag, 2006, pp. 37-53.

[10] J. Kim, A. Biryukov, B. Preneel, S. Hong" On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1," Proceedings SCN'06, LNCS 4116, R. De Prisco and M. Yung, Eds, Springer-Verlag, 2006, pp. 242-256.

[11] P. Fouque, G. Leurent, P. Q. Nguyen" Full Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5," Proceedings CRYPTO'07, LNCS 4622, A. Menezes, Ed, Springer-Verlag, 2007, pp. 13-30.

[12] L. Wang, K. Ohta, N. Kunihiro" New Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5," Proceedings EUROCRYPT'08, LNCS 4965, N. Smart, Ed, Springer-Verlag, 2008, pp. 237-253.

[13] X. Wang, H. Yu, W. Wang, H. Zhang, T. Zhan" Cryptanalysis on HMAC/NMAC-MD5 and MD5-MAC," Proceedings EUROCRYPT'09, LNCS 5479, A. Joux, Ed, Springer-Verlag, 2009, pp. 121-133.

中 华 人 民 共 和 国
国 家 标 准
信息技术 安全技术 消息鉴别码
第 2 部分：采用专用杂凑函数的机制
GB/T 15852.2—2012
*
中国标准出版社出版发行
北京市朝阳区和平里西街甲 2 号(100013)
北京市西城区三里河北街 16 号(100045)
网址 www.spc.net.cn
总编室：(010)64275323 发行中心：(010)51780235
读者服务部：(010)68523946
中国标准出版社秦皇岛印刷厂印刷
各地新华书店经销
*
开本 880×1230 1/16 印张 1.75 字数 47 千字
2013 年 5 月第一版 2013 年 5 月第一次印刷
*
书号：155066·1-46892 定价 27.00 元

GB/T 15852.2-2012

打印日期：2013年5月14日 F009