NIST Special Publication 800-38B

Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication

Morris Dworkin

C O M P U T E R S E C U R I T Y

NIST Special Publication 800-38B

Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication

Morris Dworkin

C O M P U T E R S E C U R I T Y

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

May 2005

U.S. Department of Commerce
Carlos M. Gutierrez, Secretary

Technology Administration
Phillip J. Bond, Under Secretary of Commerce for Technology

National Institute of Standards and Technology
Hratch G. Semerjian, Acting Director

Reports on Information Security Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of technical, physical, administrative, and management standards and guidelines for the cost-effective security and privacy of sensitive unclassified information in Federal computer systems. This Special Publication 800-series reports on ITL's research, guidance, and outreach efforts in computer security, and its collaborative activities with industry, government, and academic organizations.

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

National Institute of Standards and Technology Special Publication 800-38B
Natl. Inst. Stand. Technol. Spec. Publ. 800-38B, 23 pages (May 2005)
CODEN: NSPUE2

Abstract

This Recommendation specifies a message authentication code (MAC) algorithm based on a symmetric key block cipher. This block cipher-based MAC algorithm, called CMAC, may be used to provide assurance of the authenticity and, hence, the integrity of binary data.

KEY WORDS: authentication; block cipher; cryptography; information security; integrity; message authentication code; mode of operation.

Table of Contents

Figures

1 Purpose

This publication is the second Part in a series of Recommendations regarding modes of operation of symmetric key block ciphers.

2 Authority

This document has been developed by the National Institute of Standards and Technology (NIST) in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets, but such standards and guidelines shall not apply to national security systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), Securing Agency Information Systems, as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided in A-130, Appendix III.

This guideline has been prepared for use by federal agencies. It may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright. (Attribution would be appreciated by NIST.)

Nothing in this document should be taken to contradict standards and guidelines made mandatory and binding on federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other federal official.

Conformance testing for implementations of the mode of operation that is specified in this Part of the Recommendation will be conducted within the framework of the Cryptographic Module Validation Program (CMVP), a joint effort of NIST and the Communications Security Establishment of the Government of Canada. An implementation of a mode of operation must adhere to the requirements in this Recommendation in order to be validated under the CMVP. The requirements of this Recommendation are indicated by the word "shall."

3 Introduction

This Recommendation specifies a message authentication code (MAC) algorithm that is based on a symmetric key block cipher. This cipher-based MAC is abbreviated CMAC, analogous to the abbreviation for the hash function-based MAC, HMAC, that is standardized in FIPS Pub. 198 [4]. CMAC may be appropriate for information systems in which an approved block cipher is more readily available than an approved hash function.

The basic Cipher Block Chaining MAC algorithm (CBC-MAC) has security deficiencies [9]. The core of the CMAC algorithm is a variation of CBC-MAC that Black and Rogaway proposed and analyzed under the name XCBC in Ref. [2] and submitted to NIST in Ref. [1]. The XCBC

algorithm efficiently addresses the security deficiencies of CBC-MAC. Iwata and Kurosawa proposed an improvement of XCBC and named the resulting algorithm One-Key CBC-MAC (OMAC) in Ref. [6] and in Ref. [5], their initial submission to NIST; they later submitted OMAC1 [7], a refinement of OMAC, and additional security analysis [8]. The OMAC1 variation efficiently reduces the key size of XCBC. CMAC is equivalent to OMAC1.

Because CMAC is based on an approved symmetric key block cipher, such as the Advanced Encryption Standard (AES) algorithm that is specified in Federal Information Processing Standard (FIPS) Pub. 197 [3], CMAC can be considered a mode of operation of the block cipher. CMAC is also an approved mode of the Triple Data Encryption Algorithm (TDEA) [10]; however, as discussed in Appendix B, the recommended default message span for TDEA is much more restrictive than for the AES algorithm, due to the smaller block size of TDEA.

CMAC, like any well-designed MAC algorithm, provides stronger assurance of data integrity than a checksum or an error detecting code. The verification of a checksum or an error detecting code is designed to detect only accidental modifications of the data, while CMAC is designed to detect intentional, unauthorized modifications of the data, as well as accidental modifications.

4 Definitions, Abbreviations, and Symbols

4.1 Definitions and Abbreviations

AES	Advanced Encryption Standard.
Approved	FIPS approved or NIST recommended: an algorithm or technique that is either 1) specified in a FIPS or a NIST Recommendation, or 2) adopted in a FIPS or a NIST Recommendation.
Authenticity	The property that data originated from its purported source.
Bit	A binary digit: 0 or 1.
Bit String	A finite, ordered sequence of bits.
Block	For a given block cipher, a bit string whose length is the block size of the block cipher.
Block Cipher	An algorithm for a parameterized family of permutations on bit strings of a fixed length.
Block Size	For a given block cipher, the fixed length of the input (or output) bit strings.
CBC	Cipher Block Chaining.

Collision	For a given function, a pair of distinct input values that yield the same output value.
Exclusive-OR	The bitwise addition, modulo 2, of two bit strings of equal length.
FIPS	Federal Information Processing Standard.
Forward Cipher Function	A permutation on blocks that is determined by the choice of a key for a given block cipher.
Integrity	The property that received data has not been altered.
Inverse Cipher Function	The inverse function of the forward cipher function for a given block cipher key.
Key (Block Cipher Key)	The parameter of the block cipher that determines the selection of the forward cipher function from the family of permutations.
Least Significant Bit(s)	The right-most bit(s) of a bit string.
Message Authentication Code (MAC)	A bit string of fixed length, computed by a MAC generation algorithm, that is used to establish the authenticity and, hence, the integrity of a message.
MAC Generation (Generation)	An algorithm that computes a MAC from a message and a key.
MAC Verification (Verification)	An algorithm that verifies if a purported MAC is valid for a given message and key.
Mode of Operation (Mode)	An algorithm for the cryptographic transformation of data that features a symmetric key block cipher.
Most Significant Bit(s)	The left-most bit(s) of a bit string.
NIST	National Institute of Standards and Technology.
Permutation	An invertible function.
Subkey	A secret string that is derived from the key.
Subkey Generation	An algorithm that derives subkeys from a key.
TDEA	Triple Data Encryption Algorithm.

4.2 Symbols

4.2.1 Variables

b	The bit length of a block.
R_b	The constant string for subkey generation for a cipher with block size b.
K	The block cipher key.
K1	The first subkey.
K2	The second subkey.
Key1	The first component of a TDEA key.
Key2	The second component of a TDEA key.
Key3	The third component of a TDEA key.
M	The message.
M_i	The ith block of the formatted message.
M_n^*	The final block, possibly a partial block, of the formatted message.
Mlen	The bit length of the message.
n	The number of blocks in the formatted message.
T	The MAC.
Tlen	The bit length of the MAC.

4.2.2 Operations and Functions

$\lceil x \rceil$	The least integer that is not less than the real number x.
$X \| Y$	The concatenation of two bit strings X and Y.
$X \oplus Y$	The bitwise exclusive-OR of two bit strings X and Y of the same length.
$\text{CIPH}_K(X)$	The output of the forward cipher function of the block cipher under the key K applied to the block X.
$\text{LSB}_s(X)$	The bit string consisting of the s right-most bits of the bit string X.

$MSB_s(X)$ The bit string consisting of the s left-most bits of the bit string X.

$X << 1$ The bit string that results from discarding the leftmost bit of the bit string X and appending a '0' bit on the right.

$\lg(x)$ The base 2 logarithm of the positive real number x.

0^s The bit string that consists of s '0' bits.

5 Preliminaries

The elements of CMAC and the associated notation are introduced in the five sections below. Examples of operations and functions are given in Sec. 5.1. The underlying block cipher and key are discussed in Sec. 5.2. The two subkeys that are derived from the key are discussed in Sec. 5.3. MAC generation and verification are discussed in Sec. 5.4. The input and output data for MAC generation are discussed in Sec. 5.5.

5.1 Examples of Operations and Functions

Given a positive integer s, 0^s denotes the string that consists of s '0' bits. For example, $0^8 =$ 00000000.

Given a real number x, the ceiling function, denoted $\lceil x \rceil$, is the least integer that is not less than x. For example, $\lceil 2.1 \rceil = 3$, and $\lceil 4 \rceil = 4$.

The concatenation operation on bit strings is denoted ||; for example, 001 || 10111 = 00110111.

Given bit strings of equal length, the exclusive-OR operation, denoted $\oplus$, specifies the addition, modulo 2, of the bits in each bit position, i.e., without carries. For example, 10011 $\oplus$ 10101 = 00110.

Given a bit string X, the functions $LSB_s(X)$ and $MSB_s(X)$ return the s least significant (i.e., right-most) bits and the s most significant (i.e., left-most) bits, respectively, of X. For example, $LSB_3(111011010) = 010$, and $MSB_4(111011010) = 1110$.

Given a bit string X that consists of $Xlen$ bits, the (single) left-shift function, denoted $X << 1$, is $LSB_{Xlen}(X \| 0)$. For example, $1101110 << 1 = 1011100$.

Given a positive real number x, its base 2 logarithm is denoted $\lg(x)$. For example, $\lg(2^{10}) = 10$.

5.2 Block Cipher

The CMAC algorithm depends on the choice of an underlying symmetric key block cipher. The CMAC algorithm is thus a mode of operation (a mode, for short) of the block cipher. The CMAC key is the block cipher key (the key, for short).

For any given key, the underlying block cipher of the mode consists of two functions that are inverses of each other. The choice of the block cipher includes the designation of one of the two functions of the block cipher as the forward function/transformation, and the other as the inverse function, as in the specifications of the AES algorithm and TDEA in Ref. [3] and Ref. [10], respectively. The CMAC mode does not employ the inverse function.

The forward cipher function is a permutation on bit strings of a fixed length; the strings are called blocks. The bit length of a block is denoted b, and the length of a block is called the block size. For the AES algorithm, $b = 128$; for TDEA, $b = 64$. The key is denoted K, and the resulting forward cipher function of the block cipher is denoted CIPH_K.

The underlying block cipher shall be approved, and the key shall be generated uniformly at random, or close to uniformly at random, i.e., so that each possible key is (nearly) equally likely to be generated. The key shall be secret and shall be used exclusively for the CMAC mode of the chosen block cipher. The message span of the key is discussed in Appendix B. To fulfill the requirements on the key, the key should be established among the parties to the information within an approved key management structure; the details of the establishment and management of keys are outside the scope of this Recommendation.

5.3 Subkeys

The block cipher key is used to derive two additional secret values, called the subkeys, denoted *K1* and *K2*. The length of each subkey is the block size. The subkeys are fixed for any invocation of CMAC with the given key. Consequently, the subkeys may be precomputed and stored with the key for repeated use; alternatively, the subkeys may be computed anew for each invocation.

Any intermediate value in the computation of the subkey, in particular, $\text{CIPH}_K(0^b)$, shall also be secret. This requirement precludes the system in which CMAC is implemented from using this intermediate value publicly for some other purpose, for example, as an unpredictable value or as an integrity check value on the key.

One of the elements of the subkey generation process is a bit string, denoted R_b, that is completely determined by the number of bits in a block. In particular, for the two block sizes of the currently approved block ciphers, $R_{128} = 0^{120}10000111$, and $R_{64} = 0^{59}11011$.

In general, R_b is a representation of a certain irreducible binary polynomial of degree b, namely, the lexicographically first among all such polynomials with the minimum possible number of nonzero terms. If this polynomial is expressed as $u^b+c_{b-1}u^{b-1}+...+c_2u^2+c_1u+c_0$, where the coefficients $c_{b-1}, c_{b-2}, ..., c_2, c_1, c_0$ are either 0 or 1, then R_b is the bit string $c_{b-1}c_{b-2}...c_2c_1c_0$.

5.4 MAC Generation and Verification

As for any MAC algorithm, an authorized party applies the MAC generation process to the data to be authenticated to produce a MAC for the data. Subsequently, any authorized party can apply the verification process to the received data and the received MAC. Successful verification provides assurance of data authenticity, as discussed in Appendix A, and, hence, of

integrity.

5.5 Input and Output Data

For a given block cipher and key, the input to the MAC generation function is a bit string called the message, denoted *M*. The bit length of *M* is denoted *Mlen*. The value of *Mlen* is not an essential input for the MAC generation algorithm if the implementation has some other means of identifying the last block in the partition of the message, as discussed in Sec. 6.2. Thus, in such a case, the computation of the MAC may begin "on-line" before the entire message is available. In principle, there is no restriction on the lengths of messages. In practice, however, the system in which CMAC is implemented may restrict the length of the input messages to the MAC generation function.

The output of the MAC generation function is a bit string called the MAC, denoted *T*. The length of *T*, denoted *Tlen*, is a parameter that shall be fixed for all invocations of CMAC with the given key. The requirements for the selection of *Tlen* are given in Appendix A.

6 CMAC Specification

Subkey generation, MAC generation, and MAC verification are specified in Sections 6.1, 6.2, and 6.3 below. The specifications include the inputs, the outputs, a suggested notation for the function, the steps, and a summary; for MAC generation, a diagram is also given. The inputs that are typically fixed across many invocations of CMAC are called the prerequisites. The prerequisites and the other inputs shall meet the requirements in Sec. 5. The suggested notation does not include the block cipher.

6.1 Subkey Generation

The following is a specification of the subkey generation process of CMAC:

Prerequisites:
block cipher CIPH with block size *b*;
key *K*.

Output:
subkeys *K1*, *K2*.

Suggested Notation:
SUBK(*K*).

Steps:
1. Let $L = \text{CIPH}_K(0^b)$.
2. If $\text{MSB}_1(L) = 0$, then $K1 = L << 1$;
   Else $K1 = (L << 1) \oplus R_b$; see Sec. 5.3 for the definition of R_b.
3. If $\text{MSB}_1(K1) = 0$, then $K2 = K1 << 1$;
   Else $K2 = (K1 << 1) \oplus R_b$.

4. Return *K1*, *K2*.

In Step 1, the block cipher is applied to the block that consists entirely of '0' bits. In Step 2, the first subkey is derived from the resulting string by a left shift of one bit, and, conditionally, by XORing a constant that depends on the block size. In Step 3, the second subkey is derived in the same manner from the first subkey.[1] As discussed in Sec. 5.3, any intermediate value in the computation of the subkey, in particular, $\text{CIPH}_K(0^b)$, shall be secret.

6.2 MAC Generation

The following is a specification of the MAC generation process of CMAC:

Prerequisites:
block cipher CIPH with block size *b*;
key *K*;
MAC length parameter *Tlen*.

Input:
message *M* of bit length *Mlen.*

Output:
MAC *T* of bit length *Tlen.*

Suggested Notation:
CMAC(*K*, *M*, *Tlen*) or, if *Tlen* is understood from the context, CMAC(*K*, *M*).

Steps:
1. Apply the subkey generation process in Sec. 6.1 to *K* to produce *K1* and *K2*.
2. If *Mlen* = 0, let $n = 1$; else, let $n = \lceil Mlen/b \rceil$.
3. Let $M_1, M_2, \ldots, M_{n-1}, M_n^*$ denote the unique sequence of bit strings such that $M = M_1 \| M_2 \| \ldots \| M_{n-1} \| M_n^*$, where $M_1, M_2, \ldots, M_{n-1}$ are complete blocks.[2]
4. If M_n^* is a complete block, let $M_n = K1 \oplus M_n^*$; else, let $M_n = K2 \oplus (M_n^* \| 10^j)$, where $j = nb\text{-}Mlen\text{-}1$.
5. Let $C_0 = 0^b$.
6. For $i = 1$ to n, let $C_i = \text{CIPH}_K(C_{i-1} \oplus M_i)$.
7. Let $T = \text{MSB}_{Tlen}(C_n)$.
8. Return *T.*

In Step 1, the subkeys are generated from the key. In Steps 2–4, the input message is formatted into a sequence of complete blocks in which the final block has been masked by a subkey. There are two cases:

[1] As detailed in Ref. [5], the generation of *K1* and *K2* is essentially equivalent to multiplication by u and u^2, respectively, within the Galois field that is determined by the irreducible polynomial that is represented by R_b, which is discussed in Sec. 5.3.
[2] Consequently, if $Mlen \le b$, then $M = M_1^*$.

- If the message length is a positive multiple of the block size, then the message is partitioned into complete blocks. The final block is masked with the first subkey; in other words, the final block in the partition is replaced with the exclusive-OR of the final block with the first subkey. The resulting sequence of blocks is the formatted message.

- If the message length is not a positive multiple of the block size, then the message is partitioned into complete blocks to the greatest extent possible, i.e., into a sequence of complete blocks followed by a final bit string whose length is less than the block size. A padding string is appended to this final bit string, in particular, a single '1' bit followed by the minimum number of '0' bits, possibly none, that are necessary to form a complete block. The complete final block is masked, as described in the previous bullet, with the second subkey. The resulting sequence of blocks is the formatted message.

In Steps 5 and 6, the cipher block chaining (CBC) technique, with the zero block as the initialization vector, is applied to the formatted message. In Steps 7 and 8, the final CBC output block is truncated according to the MAC length parameter that is associated with the key, and the result is returned as the MAC.

Equivalent sets of steps, i.e., procedures that yield the correct output from the same input, are permitted. For example, it is not necessary to complete the formatting of the entire message (Steps 3 and 4) prior to the cipher block chaining (Steps 5 and 6). Instead, the iterations of Step 5 may be executed "on the fly," i.e., on each successive block of the message as soon as it is available for processing. Step 4 may be delayed until the final bit string in the partition is available; the appropriate case, and value of j, if necessary, can be determined from the length of the final bit string. In such an implementation, the determination in Step 2 of the total number of blocks in the formatted message may be omitted, assuming that the implementation has another way to identify the final string in the partition.

Similarly, the subkeys need not be computed anew for each invocation of CMAC with a given key; instead, they may be precomputed and stored along with the key as algorithm inputs.

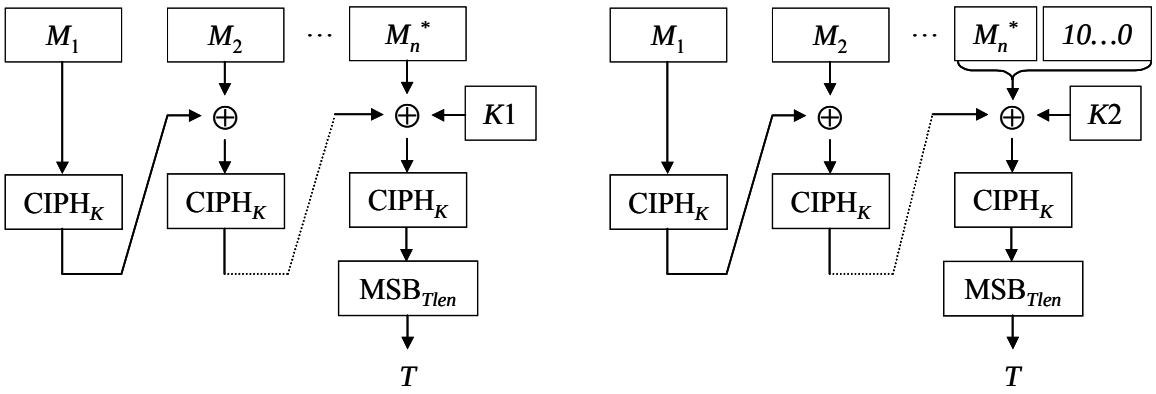

Figure 1: Illustration of the two cases of MAC Generation.

The two cases of MAC Generation are illustrated in Figure 1 above. On the left is the case where the message length is a positive multiple of the block size; on the right is the case where the message length is not a positive multiple of the block length.

6.3 MAC Verification

The following is a specification of the MAC verification process of CMAC:

Prerequisites:
block cipher CIPH with block size *b*;
key *K*;
subkeys *K1*, *K2*;
MAC length *Tlen*.

Input:
message *M* of bit length *Mlen*;
received MAC *T'*.

Output:
VALID or INVALID.

Suggested Notation:
VER(*K*, *M*, *T'*).

Steps:
1. Apply the MAC generation process in Sec. 6.2 to *M* to produce *T*.
2. If *T* = *T'*, return VALID; else, return INVALID.

In Step 1, the MAC generation process in Sec. 6.2 is applied to the message, and, in Step 2, the resulting MAC is compared with the received MAC to determine its validity.

Appendix A: Length of the MAC

The length, *Tlen*, of the MAC is an important security parameter. The role of this parameter in resisting guessing attacks is outlined in Sec. A.1, and guidance in the selection of *Tlen* is given in Sec. A.2.

A.1 Assurance Against Guessing Attacks

The verification process determines whether the purported MAC on a message is the valid output of the MAC generation process applied to the message. The output of the MAC verification determines the assurance that the receiver of the message obtains:

- If the output is INVALID, then the message is definitely not authentic, i.e., it did not originate from a source that executed the generation process on the message to produce the purported MAC.

- If the output is VALID, then the design of the mode provides assurance that the message is authentic and, hence, was not corrupted in transit; however, this assurance, as for any MAC algorithm, is not absolute.

In the second case, an attacker, i.e., a party without access to the key or to the MAC generation process, may have simply guessed the correct MAC for the message. In particular, if the attacker selects a MAC at random from the set of strings of length *Tlen* bits, then the probability is 1 in 2^{Tlen} that the MAC will be valid. Consequently, larger values of *Tlen* provide greater protection against such an event. Of course, an attacker may attempt to systematically guess many different MACs for a message, or for different messages, and thereby increase the probability that one (or more) of them will be accepted as valid. For this reason, a system should limit the number of unsuccessful verification attempts for each key.

A.2 Selection of the MAC Length

Larger values of *Tlen* provide greater assurance against guessing attacks. The performance tradeoff is that larger values of *Tlen* require more bandwidth/storage for the MAC.

For most applications, a value for *Tlen* that is at least 64 should provide sufficient protection against guessing attacks. A value of *Tlen* that is less than 64 shall only be used in conjunction with a careful analysis of the risks of accepting an inauthentic message as authentic.

In particular, a value of *Tlen* smaller than 64 should not be used unless the controlling protocol or system sufficiently restricts the number of times that the verification process can return INVALID, across all implementations with any given key. For example, the short duration of a session or, more generally, the low bandwidth of the communication channel may preclude many repeated trials.

This guidance can be quantified in terms of the following two bounds: 1) the highest acceptable probability for an inauthentic message to pass the verification process, and 2) a limit on the number of times that the output is the error message INVALID before the key is retired, across all implementations of the verification process for the key. Given estimates of these quantities, denoted *Risk* and *MaxInvalids*, respectively, *Tlen* should satisfy the following inequality:

$$Tlen \geq \lg(MaxInvalids / Risk) .$$

For example, suppose that the MAC verification process(es) within a system will not output INVALID for more than 1024 messages before the key is retired (i.e., $MaxInvalids = 2^{10}$), and that the users can tolerate about a one in a million chance that the system will accept an inauthentic message (i.e., $Risk = 2^{-20}$). In this case, any value of *Tlen* that is greater than or equal to 30 satisfies the inequality.

Appendix B: Message Span of the Key

The message span of a key is the total number of messages for which MACs are generated across all implementations of CMAC with that key. The message span of the key affects the security of the system against attacks that are based on the detection of a pair of distinct messages with the same MAC before its truncation[3]. Such a pair is called a collision[4] in this appendix. As with other block cipher-based MAC algorithms, an attacker may be able to exploit a collision to produce the valid MAC for a new message, the content of which may be largely of the attacker's choosing. Such an event would be a fundamental breach of the expected authentication assurance.

In principle, collisions must exist because there are many more possible messages than MACs; in practice, however, collisions may not occur among the messages for which MACs are actually generated during the lifetime of the key. The probability that at least one collision will occur depends mostly on the message span of the key relative to the block size, b, of the underlying block cipher. For example, a collision is expected to exist among a set of $2^{b/2}$ arbitrary messages; in other words, 2^{64} messages for the AES algorithm, and 2^{32} messages for TDEA. This property was one of the motivations to develop the AES with a block size of 128 bits.

For any system in which CMAC is implemented, the risk that an attacker can detect and exploit a collision shall be limited to a level that is appropriate to the value of the data. A simple and prudent method to achieve this goal is to establish and enforce an appropriate limit on the message span of any CMAC key, which in turn limits the probability that a collision will even occur. For general-purpose applications, the default recommendation is to limit the key to no more than 2^{48} messages when the block size of the underlying block cipher is 128 bits, as with the AES algorithm, and 2^{21} messages when the block size is 64 bits, as with TDEA. Within these limits, the probability that a collision will occur is expected to be less than one in a billion for the AES algorithm, and less than one in a million for TDEA.

For applications where higher confidence in the security is required, the message span of a key may be measured in terms of the total number of message blocks. The recommendation in this case is to limit the key to no more than 2^{48} message blocks (2^{22} Gbytes) when the block size is 128 bits, and 2^{21} message blocks (16 Mbytes) when the block size is 64 bits. Within these limits, the probability that a collision will occur is proved to be less than one in a billion for the AES algorithm, and less than one in a million for TDEA, assuming that the underlying block cipher has no weakness, as modeled in Ref. [6].

In some cases, a limit on the message span of a key may be established and enforced within a key management infrastructure by an appropriate constraint on the time span during which the key remains in effect, i.e., its cryptoperiod.

[3] The MAC before truncation is denoted C_m in Sec. 6.2.

[4] The standard definition of a collision, in Ref. [9], for example, is more general: for a given function, a collision is a pair of distinct input values that yield the same output value.

Appendix C: Protection Against Replay of Messages

As described in Appendix A, the successful verification of a MAC for a message gives assurance that the source of the message executed the MAC generation algorithm to create the MAC; however, the party that presented the message and MAC for verification may not be the original source of the message. Therefore, the CMAC algorithm does not inherently prevent an attacker from intercepting a legitimate message and its MAC and "replaying" them for verification at a later time, for example, in an attempt to impersonate a party that has access to the key. In some protocols an attacker may even be able to present to a verifier a message-MAC pair that the verifier itself generated earlier in the protocol.

The controlling protocol or application may protect against such an event by incorporating certain identifying information into the initial bits of every message. Examples of such information include a sequential message number, a timestamp, or a nonce. Upon successful verification of the message, this information may provide a means for the detection of replayed messages, out-of-sequence messages, or missing messages.

Appendix D: Examples

In this appendix, twenty examples are provided for the MAC generation process. The underlying block cipher is either the AES algorithm or TDEA. A block cipher key is fixed for each of the currently allowed key sizes, i.e., AES-128, AES-192, AES-256, two key TDEA, and three key TDEA. For each key, the generation of the associated subkeys is given, followed by four examples of MAC generation with the key. The messages in each set of examples are derived by truncating a common fixed string of 64 bytes.

All strings are represented in hexadecimal notation, with a space (or a new line) inserted every 8 symbols, for readability. As before, *K1* and *K2* denote the subkeys, *M* denotes the message, and *T* denotes the MAC. For the AES algorithm examples, *Tlen* is 128, i.e., 32 hexadecimal symbols, and *K* denotes the key. For the TDEA examples, *Tlen* is 64, i.e., 16 hexadecimal symbols, and the key, *K*, is the ordered triple of strings, (*Key1*, *Key2*, *Key3*), consistent with Ref. [10]. For two key TDEA, *Key1* = *Key3*.

D.1 AES-128

For Examples 1–4 below, the block cipher is the AES algorithm with the following 128 bit key:

K 2b7e1516 28aed2a6 abf71588 09cf4f3c.

Subkey Generation

$CIPH_K(0^{128})$ 7df76b0c 1ab899b3 3e42f047 b91b546f

K1 fbeed618 35713366 7c85e08f 7236a8de

K2 f7ddac30 6ae266cc f90bc11e e46d513b

Example 1: *Mlen* 0

M <empty string>

T bb1d6929 e9593728 7fa37d12 9b756746

Example 2: *Mlen* = 128

M 6bc1bee2 2e409f96 e93d7e11 7393172a

T 070a16b4 6b4d4144 f79bdd9d d04a287c

Example 3: *Mlen* = 320

M 6bc1bee2 2e409f96 e93d7e11 7393172a
ae2d8a57 1e03ac9c 9eb76fac 45af8e51
30c81c46 a35ce411

T dfa66747 de9ae630 30ca3261 1497c827

Example 4: *Mlen* = 512

M 6bc1bee2 2e409f96 e93d7e11 7393172a
ae2d8a57 1e03ac9c 9eb76fac 45af8e51
30c81c46 a35ce411 e5fbc119 1a0a52ef

	f69f2445 df4f9b17 ad2b417b e66c3710
T	51f0bebf 7e3b9d92 fc497417 79363cfe

D.2 AES-192

For Examples 5–8 below, the block cipher is the AES algorithm with the following 192 bit key:

K	8e73b0f7 da0e6452 c810f32b 809079e5
	62f8ead2 522c6b7b.

Subkey Generation

$CIPH_K(0^{128})$	22452d8e 49a8a593 9f7321ce ea6d514b
K1	448a5b1c 93514b27 3ee6439d d4daa296
K2	8914b639 26a2964e 7dcc873b a9b5452c

Example 5: *Mlen* = 0

M	<empty string>
T	d17ddf46 adaacde5 31cac483 de7a9367

Example 6: *Mlen* = 128

M	6bc1bee2 2e409f96 e93d7e11 7393172a
T	9e99a7bf 31e71090 0662f65e 617c5184

Example 7: *Mlen* = 320

M	6bc1bee2 2e409f96 e93d7e11 7393172a
	ae2d8a57 1e03ac9c 9eb76fac 45af8e51
	30c81c46 a35ce411
T	8a1de5be 2eb31aad 089a82e6 ee908b0e

Example 8: *Mlen* = 512

M	6bc1bee2 2e409f96 e93d7e11 7393172a
	ae2d8a57 1e03ac9c 9eb76fac 45af8e51
	30c81c46 a35ce411 e5fbc119 1a0a52ef
	f69f2445 df4f9b17 ad2b417b e66c3710
T	a1d5df0e ed790f79 4d775896 59f39a11

D.3 AES-256

For Examples 9–12 below, the block cipher is the AES algorithm with the following 256 bit key:

K	603deb10 15ca71be 2b73aef0 857d7781
	1f352c07 3b6108d7 2d9810a3 0914dff4.

Subkey Generation

$CIPH_K(0^{128})$	e568f681 94cf76d6 174d4cc0 4310a854
K1	cad1ed03 299eedac 2e9a9980 8621502f
K2	95a3da06 533ddb58 5d353301 0c42a0d9

Example 9: *Mlen* = 0

M	<empty string>
T	028962f6 1b7bf89e fc6b551f 4667d983

Example 10: *Mlen* = 128

M	6bc1bee2 2e409f96 e93d7e11 7393172a
T	28a7023f 452e8f82 bd4bf28d 8c37c35c

Example 11: *Mlen* = 320

M	6bc1bee2 2e409f96 e93d7e11 7393172a
	ae2d8a57 1e03ac9c 9eb76fac 45af8e51
	30c81c46 a35ce411
T	aaf3d8f1 de5640c2 32f5b169 b9c911e6

Example 12: *Mlen* = 512

M	6bc1bee2 2e409f96 e93d7e11 7393172a
	ae2d8a57 1e03ac9c 9eb76fac 45af8e51
	30c81c46 a35ce411 e5fbc119 1a0a52ef
	f69f2445 df4f9b17 ad2b417b e66c3710
T	e1992190 549f6ed5 696a2c05 6c315410

D.4 Three Key TDEA

For Examples 13-16 below, the block cipher is three key TDEA with the following key:

Key1	8aa83bf8 cbda1062
Key2	0bc1bf19 fbb6cd58
Key3	bc313d4a 371ca8b5

Subkey Generation

$CIPH_K(0^{64})$	c8cc74e9 8a7329a2
K1	9198e9d3 14e6535f
K2	2331d3a6 29cca6a5

Example 13: *Mlen* = 0

M	<empty string>
T	b7a688e1 22ffaf95

Example 14: *Mlen* = 64

M	6bc1bee2 2e409f96
T	b7a688e1 22ffaf95

Example 15: *Mlen* = 160

M	6bc1bee2 2e409f96 e93d7e11 7393172a

```
                ae2d8a57
T               d32bcebe 43d23d80
```

Example 16: *Mlen* = 256

```
M               6bc1bee2 2e409f96 e93d7e11 7393172a
                ae2d8a57 1e03ac9c 9eb76fac 45af8e51
T               33e6b109 2400eae5
```

D.5 Two Key TDEA

For Examples 17-20 below, the block cipher is two key TDEA with the following key:

```
Key1            4cf15134 a2850dd5
Key2            8a3d10ba 80570d38
Key3            4cf15134 a2850dd5
```

Subkey Generation

```
CIPH_K(0^64)         c7679b9f 6b8d7d7a
K1              8ecf373e d71afaef
K2              1d9e6e7d ae35f5c5
```

Example 17: *Mlen* = 0

```
M               <empty string>
T               bd2ebf9a 3ba00361
```

Example 18: *Mlen* = 64

```
M               6bc1bee2 2e409f96
T               bd2ebf9a 3ba00361
```

Example 19: *Mlen* = 160

```
M               6bc1bee2 2e409f96 e93d7e11 7393172a
                ae2d8a57
T               8ea92435 b52660e0
```

Example 20: *Mlen* = 256

```
M               6bc1bee2 2e409f96 e93d7e11 7393172a
                ae2d8a57 1e03ac9c 9eb76fac 45af8e51
T               31b1e431 dabc4eb8
```

Appendix E: Bibliography

[1] J. Black, P. Rogaway, A Suggestion for Handling Arbitrary-Length Messages with the CBC MAC, Natl. Inst. Stand. Technol. [Web page], http://csrc.nist.gov/CryptoToolkit/modes/workshop1/.

[2] J. Black, P. Rogaway, CBC MACs for arbitrary-length messages: The three-key constructions, in Advances in Cryptology—Crypto 2000, Lecture Notes in Computer Science, Vol. 1880, Mihir Bellare, ed., Springer-Verlag (2000), pp. 197–215.

[3] FIPS Publication 197, The Advanced Encryption Standard (AES), U.S. DoC/NIST, November 26, 2001.

[4] FIPS Publication 198, The Keyed-Hash Message Authentication Code, U.S. DoC/NIST, March 6, 2002.

[5] T. Iwata, K. Kurosawa, OMAC: One-Key CBC MAC, Natl. Inst. Stand. Technol. [Web page], http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/.

[6] T. Iwata, K. Kurosawa, OMAC: One-Key CBC MAC, in Fast Software Encryption, 10th International Workshop, FSE 2003, Lecture Notes in Computer Science, Vol. 2887, Thomas Johansson, ed., Springer-Verlag (2003), p.p. 129–153.

[7] T. Iwata, K. Kurosawa, OMAC: One-Key CBC MAC—Addendum, Natl. Inst. Stand. Technol. [Web page], http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/.

[8] T. Iwata, K. Kurosawa, Stronger Security Bounds for OMAC, TMAC, and XCBC, Natl. Inst. Stand. Technol. [Web page], http://csrc.nist.gov/CryptoToolkit/modes/comments/.

[9] A. Menezes, P. van Oorschot, S. Vanstone, Handbook of Applied Cryptography, CRC Press, Inc., Boca Raton (1996).

[10] NIST Special Publication 800-67 Version 1, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, May 2004, Natl. Inst. Stand. Technol. [Web page], http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf.